

For Reference

NOT TO BE TAKEN FROM THIS ROOM

Ex LIBRIS
UNIVERSITATIS
ALBERTAENSIS



THE UNIVERSITY OF ALBERTA

DESIGN AND IMPLEMENTATION OF A COMPUTER-BASED
TEACHER-AUTHORED INSTRUCTION MANAGER
(T A I M)

by



DONALD EDWARD ZARSKY

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE
OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTING SCIENCE

EDMONTON, ALBERTA

FALL, 1973

ABSTRACT

Individualized Instruction - an educational process in which each pupil encounters a different set of instructional materials and progresses at his own rate - has been and currently is the ideal of many educators. However, before this ideal can be realized today, instructors must be provided with tools for managing the many diverse requirements inherent in such an educational enterprise.

Many of the problems associated with individualized instruction are clerical problems such as updating curriculum materials, record keeping, marking tests and monitoring student progress. Since the digital computer is adept at performing clerical tasks, a project was undertaken at the University of Alberta to rationalize, describe, design, implement and evaluate a computer managed instruction system called the Teacher-Authored Instruction Manager (TAIM).

Results of the TAIM project are reported in two separate theses. The description, design and implementation of the system, from a computing scientist's point of view, is discussed by the author in this thesis, whereas the rationale, description and evaluation of the system, from an

ABSTRACT

Digitized by the Internet Archive
in 2023 with funding from
University of Alberta Library

<https://archive.org/details/Zarsky1973>

educator's viewpoint, is discussed by M. L. Westrom in a thesis entitled "The Teacher-Authored Instruction Manager (TAIM) : A Computer Managed Instruction System".

More specifically this thesis contains: a general description of the TAIM System; a breakdown of the TAIM System into its Electronic Data Processing (EDP) components; discussions on how various EDP components of TAIM were implemented on an IBM 360/67 computer running under the Michigan Terminal System (MTS) using PL/1 as the source language; detailed diagrams and accounts of the record layouts and file structures designed for and utilized by the TAIM System; the syntax and description of a command language used for interacting with TAIM; and recommendations for future implementations.

ACKNOWLEDGEMENTS

I wish to express my sincere appreciation and gratitude to Dr. S. M. Hunka, my supervisor, for his advice, guidance and patience throughout the preparation of the thesis; to Mr. M. L. Westrom for many informative discussions and valuable suggestions; and to the Division of Educational Research Services at the University of Alberta for providing the necessary financial support.

DEDICATION

Audrey and Kevin

TABLE OF CONTENTS

CHAPTER	PAGE
I. INTRODUCTION	1
II. TAIM - GENERAL DESCRIPTION	5
Introduction	5
Definitions	6
Curriculum Network	9
Student Flow	10
Logic Statements	12
Lesson Logic Statements	12
Decision Logic Statements	13
Conditional Expressions	13
Single Conditions	13
Double Conditions	14
Day (Logic) Construction	15
III. TAIM - SYSTEM DESIGN	17
Introduction	17
Interactive TAIM Subsystem	19
Mode 1 Commands	21
Mode 2 Commands	23
Mode 3 Commands	23
Command Monitor	24
Attention Interruptions	27
Decision Logic Statement Interpretation ..	27
Batch TAIM Subsystem	29
Batch Processor	31

CHAPTER	PAGE
Lesson Logic Statement Interpretation	34
IV. TAIM - FILE STRUCTURES	36
Introduction	36
The TAIM System Files	37
System File	39
Display File	42
Logic File	44
Test File	46
Student File	48
Hold File	50
Note File	52
User File	55
File Directories	57
Insertions	58
Deletions	59
Record Key Generation	59
Display, Test and Logic Files	59
Student File	60
V. TAIM - IMPLEMENTATION	62
Introduction	62
Interactive TAIM Subsystem	63
Command Table	63
Logical I/O Unit Assignments	64
Command Recognition	66
Work Module Selection	67
The EDIT Command	68

CHAPTER	PAGE
Implementing Workspaces	69
Resolving and Unresolving References	69
Batch TAIM Subsystem	71
Improving Directory Search Times	75
Illustrating the TAIM System	77
Initializing the TAIM System	77
Building a Sample Curriculum	80
Entering Student Responses	92
Sample Student Lessons	97
VI. CONCLUSION	101
Summary	101
Future Considerations	103
Chaining the Student Directory by Class ..	103
Dynamic Loading of Modules	104
Saving Directory Index Tables	106
Optimizing Record Lengths	107
Concluding Remark	107
REFERENCES	108
APPENDIX A: THE TAIM COMMAND LANGUAGE	110
APPENDIX B: DAY (LOGIC) CONSTRUCTION SYNTAX	128
APPENDIX C: RECORD LAYOUTS	132
APPENDIX D: SAMPLE SYSTEM FILE	145
APPENDIX E: CONTROL STATEMENTS FOR USING TAIM UNDER THE MICHIGAN TERMINAL SYSTEM (MTS)	159
APPENDIX F: TAIM MODULE DESCRIPTIONS	163

LIST OF TABLES

TABLE	PAGE
3.1 Summary of TAIM Commands and Subcommands	22
3.2 Initializing the Currently Active Mode	24
4.1 Record-File Assiciations	38
5.1 Interactive TAIM Subsystem Logical I/O Units ..	65
5.2 Batch TAIM Subsystem Logical I/O Units	73

LIST OF FIGURES

FIGURE	PAGE
2.1 Logic and Display File General Descriptions ...	7
2.2 Test and Student File General Descriptions	8
2.3 Curriculum Network	9
2.4 Student Network Sequencing	10
2.5 A Typical Day (Logic) Construct	15
2.6 Using the Student, Logic, Display and Test Files to Produce a Typical Student Lesson	16
3.1 The TAIM System	18
3.2 Interactive TAIM Subsystem	20
3.3 User Responsibility Levels	24
3.4 Command Monitor	25
3.5 Decision Logic Statement Interpretation	28
3.6 Batch TAIM Subsystem	30
3.7 Batch Processor	32
3.8 Lesson Logic Statement Interpretation	35
4.1 System File Record Relationships	40
4.2 Display File Record Relationships	43
4.3 Logic File Record Relationships	45
4.4 Test File Record Relationships	47
4.5 Student File Record Relationships	49
4.6 Hold File Record Relationships	51
4.7 Note File Record Relationships	53
4.8 User File Record Relationships	56

FIGURE	PAGE
5.1 Command Buffer Component Table	66
5.2 Lists Used in Reference Maintenance	70
5.3 File Directory Index Table	76
5.4 Sample Curriculum Network	80
6.1 Chaining the Student Directory by Class	105

CHAPTER I

INTRODUCTION

In the past, new approaches to the individualization of instruction, a recurring theme in educational thought, have resulted from technological developments such as the tape recorder, the teaching machine and the television monitor. Thus, with the advent of the digital computer, it was inevitable that this powerful technology would stimulate further attempts to solve the problems of individualizing instruction.

One of the most difficult aspects of a program of individualized instruction is that of management. As pointed out by Baker[1]

"A situation develops rather quickly in which each pupil is employing a different set of instructional materials, progressing at his own rate, and experiencing a unique set of successes and learning difficulties. In the center of this milieu is the classroom teacher who somehow must regulate this activity, diagnose each child's difficulties and prescribe his activities for some future period of time . . . It quickly becomes apparent that in order to operate in such a context a teacher must be provided with assistance in the management aspect of individualized instruction."

A possible solution to this problem is a computer-based instructional management system similar to what Salisbury[13] describes as being:

"... an overall system for educational management in which detailed student information, complete curriculum data and information on available resources are integrated to develop individualized programs of instruction, revise curriculum content, provide for necessary counseling and guidance and facilitate optimum educational resource management".

Such a system would be the means by which a teacher manages an educational enterprise that provides each learner with an optimum set of educational experiences.

In recent years a number of computer-based instructional management systems have been developed. Some examples are:

- (1) IMP - Instructional Management Program [2]
- (2) IMS - Instructional Management System [15]
- (3) IPI/MIS - Individually Prescribed Instruction/Management and Information System [3]
- (4) PLAN - Program for Learning in Accordance with Needs [4]
- (5) TIPS - Teaching Information Processing System [9].

From descriptions of these systems, it is apparent that they all follow the same basic model (test scoring, diagnosis, prescription, reporting) and that they all are based on a curricular approach in which educational objectives are defined in detail (Baker[1]).

Given the instructional objectives and the corresponding curriculum materials a means must be available for determining whether or not a student using them has achieved a particular objective. Typically, tests are administered during the course of instruction and resultant test scores are used as parameters in determining to what extent an objective has been mastered (diagnosis), and also in determining what tasks/objectives the student is next to perform/achieve (prescription).

Generally, diagnostic and prescriptive procedures consist of table look-up schemes in which the test score distribution is divided into several ranges, and remedial and enrichment actions are assigned "a priori" to each score interval. A student's test score is compared to the distributions; the interval in which it falls determines what task(s) the student is to perform next.

There are however, two major problems associated with the above systems:

(1) Summary response times are slow - turnaround times vary from overnight to a few days and as a result teachers receive student progress reports at inappropriate times:

"Classroom teachers typically perform lesson planning and evaluation activities at the close of each school day; [therefore], a minimum goal should be to have all reports available after the last class period (Baker[1])."

(2) Prescriptive procedures and curriculum materials are not readily modified - these are in continuous need of improvements (modifications and/or extensions). However, making improvements usually requires that teachers using a system prepare and submit written criticisms to experts who review the submissions and incorporate alterations.

" this procedure is inefficient - information is lost both when the teacher encodes (writes) the suggestions and when the expert decodes (reads) them. The amount of information transmitted will be much less than what is available (the teacher will not include all reasons for requesting a modification, nor all suggestions); and it is not fast enough (a suggestion which is accepted may not be implemented for up to a year or more). Clearly the best feedback will result if the instructor implements his own suggestions directly and immediately (Westrom [17])."

The remaining pages of this thesis will explicate the system design and implementation of an instructional manager in which the above difficulties have been overcome.

In reading this thesis, it should be kept in mind that the design of the system is based upon the following principles: (1) all or part of the system must be operable interactively; (2) response and search times must be minimal; (3) the system must be easy to use and must provide comprehensive diagnostic messages; (4) system files must be flexible and thus easily modified; and (5) the integrity of the system and its files must be preserved.

CHAPTER II

TAIM - GENERAL DESCRIPTION

2.1 Introduction

The Teacher-Authored Instruction Manager (TAIM) is an instructional assistant with the following capabilities:

- (1) storage of teacher-prepared curriculum materials and tests;
- (2) storage of teacher-prepared algorithms (algorithms define lessons and determine what instruction an individual student is to receive next);
- (3) automatic sequencing of students through the algorithms thereby disseminating curriculum materials and tests to individual students;
- (4) automatic scoring of multiple choice tests;
- (5) automatic storing of question responses and test scores;
- (6) rapid retrieval of specific information concerning student progress; and
- (7) easy modification of curriculum materials, tests, algorithms and individual student sequences.

2.2 Definitions

DISPLAY - a Display consists of a variable number of lines of textual material. The content of a Display might be a directive (read chapter six in your Mathematics text) or it might be a lengthy discussion of some concept (Euclidean Geometry). In general, Displays hold curriculum material to be presented to students.

TEST - a Test has two parts: a textual part and a key part. The textual part contains a number of multiple choice test questions, whereas the key part contains information to be used in marking question responses. Test results are used as parameters in governing the curriculum materials presented to students.

DAY - a Day is an algorithm consisting of two parts: a lesson part and a decision part. The lesson part contains statements which govern the administration of Displays and Tests to individual students, whereas the decision part contains statements used in determining what the next Day (lesson) for a particular student should be.

DIARY - for each student registered in the system there exists a Diary which contains registration data (such as name, age, sex, etc.); a history of all Displays and Tests received and the Days from which they were administered; a complete record of question responses and Test results; and

a pointer indicating which Day the student is to encounter next.

NAME - a sequence of characters, the first of which is alphabetic, used to identify a particular Display, Day, Test or Diary.

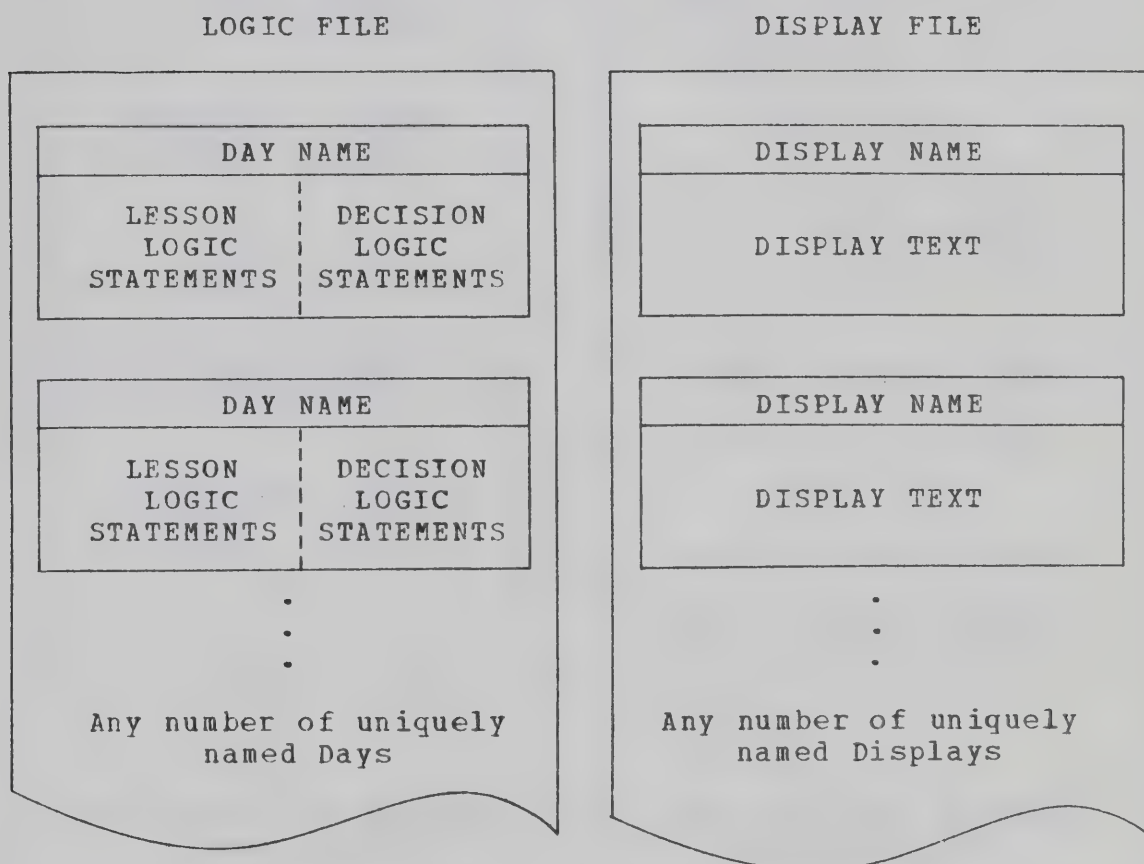


Figure 2.1 Logic and Display File General Descriptions

DISPLAY FILE - an organized collection of uniquely named Displays (see Figure 2.1).

LOGIC FILE - an organized collection of uniquely named Days.

STUDENT FILE - an organized collection of uniquely named Diaries (see Figure 2.2).

TEST FILE - an organized collection of uniquely named Tests.

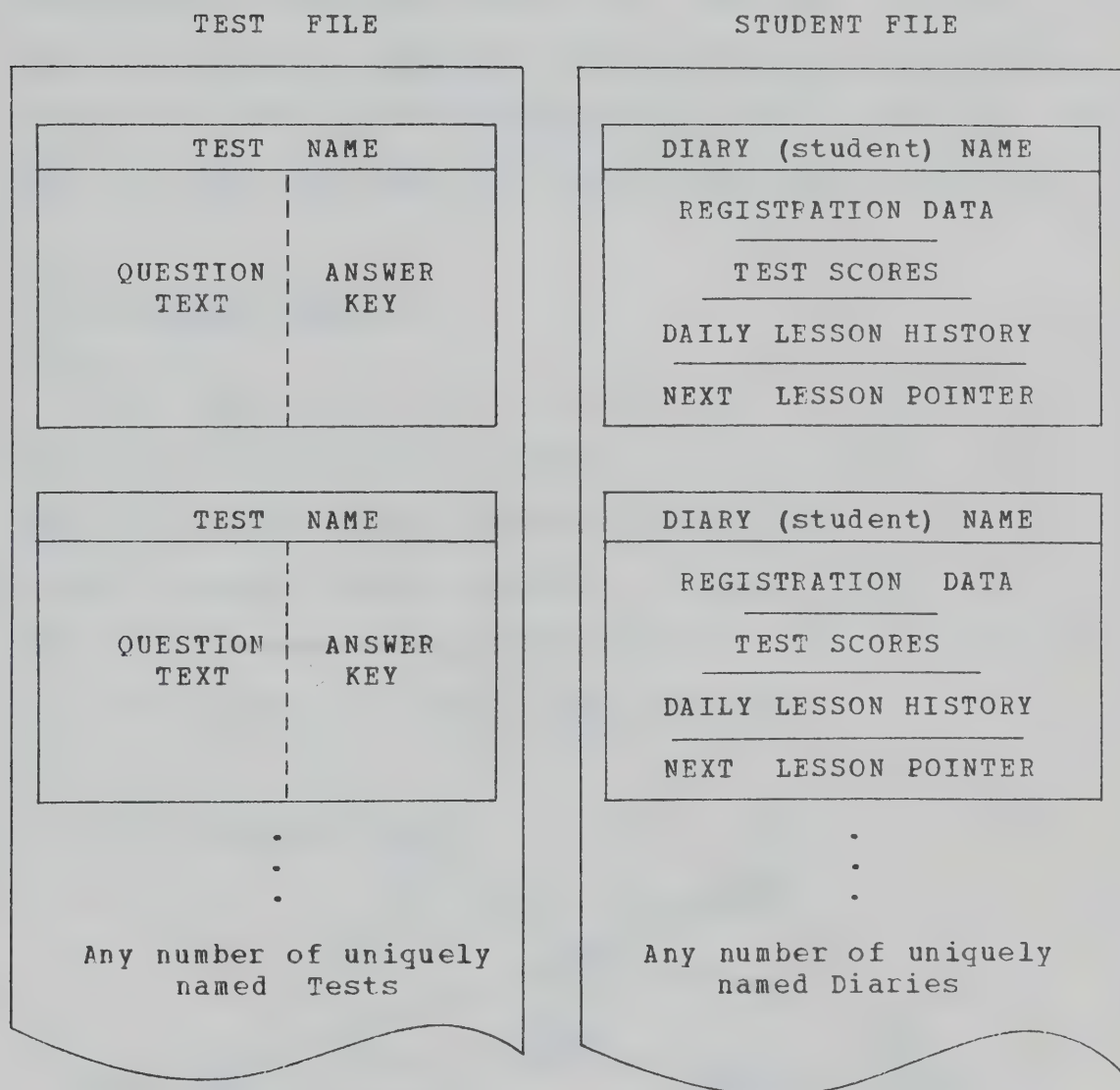


Figure 2.2 Test and Student File General Descriptions

LABEL - a qualified NAME. A name may not be used in the same file more than once. For example, there may not exist two Displays in the Display File with the name "LOGARITHMS".

However, the same identifier may be re-used in any of the other files. For example, there might exist both a Display and a Test with the name "LOGARITHMS". A reference to "LOGARITHMS" would be ambiguous without some means of indicating to which file the name applies. Therefore, a label is simply a qualified name, that is, a name which is prefixed by "D-" for the Display File, "T-" for the Test File, "L-" for the Logic File or "S-" for the Student File.

2.2 Curriculum Network

The Logic File may be thought of as a graph, where a graph is defined as being a set of two or more different "points", with certain pairs of these points joined by one or more "lines". For our purposes a graph will define a "curriculum network" where the points are Days and the lines are relationships that exist among the Days.

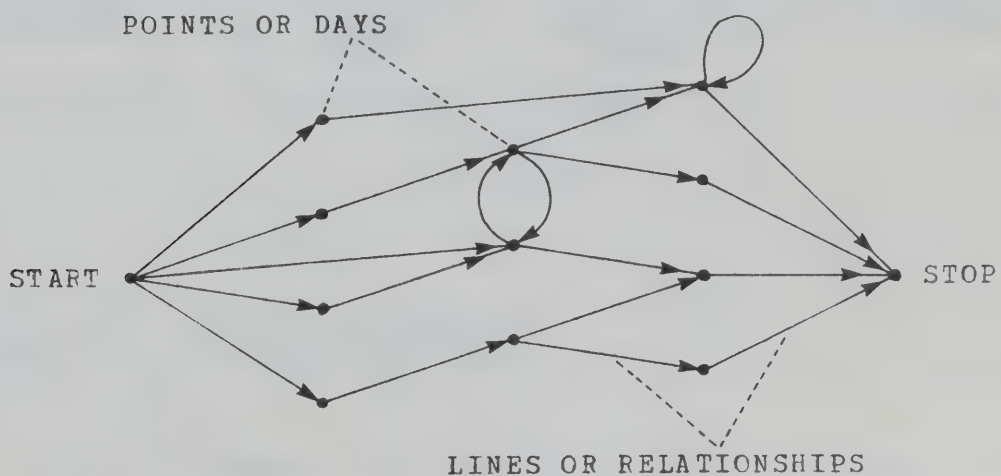


Figure 2.3 Curriculum Network

An important notion is that of students "travelling" from one part of the directed graph to another. In TAIM a path can be defined as being a sequence of Days such that each Day is in the graph and there is a relation between each consecutive pair of Days.

As registered students "flow" through the curriculum network along individual paths and at different rates, the Student File keeps an historical record of their past and present positions.

Student Flow

The sequencing of students through the curriculum network may best be described as follows:

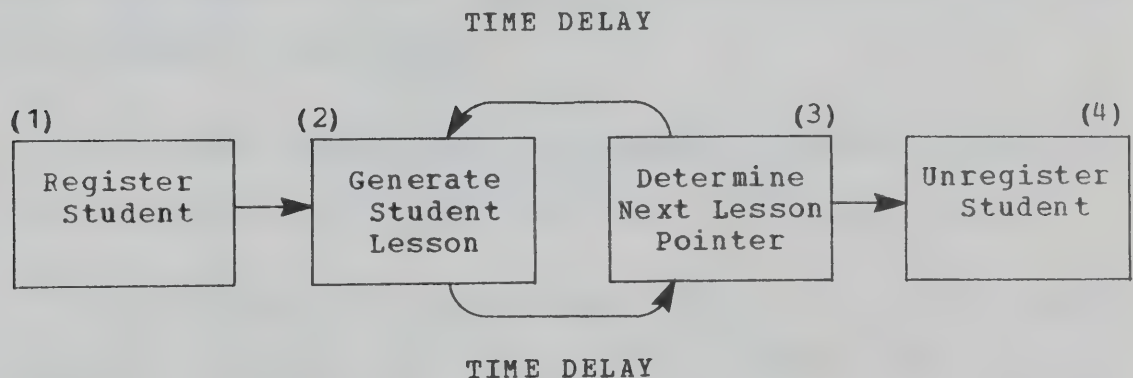


Figure 2.4 Student Sequencing

(1) A student is registered into the system and his next Day pointer is set to the first Day of the curriculum network.

(2) That night, the student's next Day pointer is retrieved, the named Day is located in the Logic File, and a lesson consisting of Display(s) and/or Test(s) is generated via interpretation of "lesson" logic statements.

(3) The following morning, the student receives his lesson printout, does the indicated work, and turns in a personalized computer card. If the lesson contained a Test, the student would have penciled his test responses on this card. The teacher, upon gathering the student response cards, enters the cards into the TAIM system. For each card entered, TAIM marks the Test if necessary according to an answer key, retrieves the student's Day in the Logic File, determines via "decision" logic statement interpretation what the student's next lesson should be and sets his next Day indicator accordingly.

(4) Steps (2) and (3) are repeated until the registered student completes the last Day in the curriculum network.

It should be noted that during the time delays indicated on Figure 2.4, the teacher, using TAIM commands described in APPENDIX A, can monitor student progress through the network, alter the students' next lesson pointers and dynamically modify the contents of Displays, Tests and Logic Statements within each Day.

2.4 Logic Statements

As was discussed earlier, a Day consists of logic statements for constructing student lesson printouts and logic statements for determining what Day the student should next encounter. Such statements will hereafter be called "Lesson Logic Statements" and "Decision Logic Statements".

Lesson Logic Statements

SHOW D-name	causes the contents of the named Display to be printed out.
TEST T-name	causes the textual portion of the named Test to be printed out.
MESG "message"	when an individual student encounters this statement, his name, the Day name, and the "message" are directed to the teacher. By placing these statements strategically, the teacher can cause the computer to warn her of situations requiring some special attention.
IF (condition) SHOW D-name	allows a Display to be conditionally printed out.
IF (condition) MESG "message"	allows a message to be conditionally directed.

Decision Logic Statements

GOTO L-name	this statement causes the student's next lesson pointer to be changed to that of the indicated Day.
WHEN (condition) GOTO L-name	if the condition is true, the student's next Day pointer becomes L-name, otherwise the next Decision Logic Statement is interpreted.

Conditional Expressions

Although many possible forms of conditions are conceivable, the current version of TAIM is limited to comparing some pre-specified threshold value with: (1) Test marks; (2) individual question responses; and (3) counts of the number of times a student has passed through a particular Day of the curriculum network.

Conditions may be either Single Conditions or Double Conditions.

Single Conditions

The Single Condition takes the form

(VALUE RELATION THRESHOLD)

where VALUE is the count of the number of times a student has encountered a specified Day, the student's total or percentage score on a particular Test, or the student's response to a particular question within a specified Test;

the RELATION is one of "EQ" (equal), "NE" (not equal), "GT" (greater than), "GE" (greater than or equal), "LT" (less than), "LE" (less than or equal); and THRESHOLD is a positive integer followed by an optional percent (%) sign or a single digit followed by a response (R) sign. The condition is true if the VALUE bears the indicated relationship to the specified THRESHOLD, false otherwise.

EXAMPLES:

If a student has encountered Day "CHAPTER:3" twice, received 12 out of 20 marks on the most recent Test "CHAP.3", and gave a response of "3" to question 1 of the same Test, then

(L-CHAPTER:3 GT 1) ----- would be true,
 (T-CHAP.3 LT 15) ----- would be true,
 (T-CHAP.3 LT 15%) ----- would be false,
 (T-CHAP.3(1) EQ 3R) ----- would be true.

Double Conditions

A Double Condition consists of two Single Conditions connected by either "AND" or "OR". If the connector is "AND", then the condition is true if both Single Conditions are true. If the connector is "OR", then the condition is true if either, or both, of the Single Conditions is true.

EXAMPLES:

(L-CHAPTER:3 EQ 3 AND T-CHAP.3 GT 50%) --- would be false,
 (T-CHAP.3 GT 10 OR T-CHAP.3(1) NE 2R) ---- would be true.

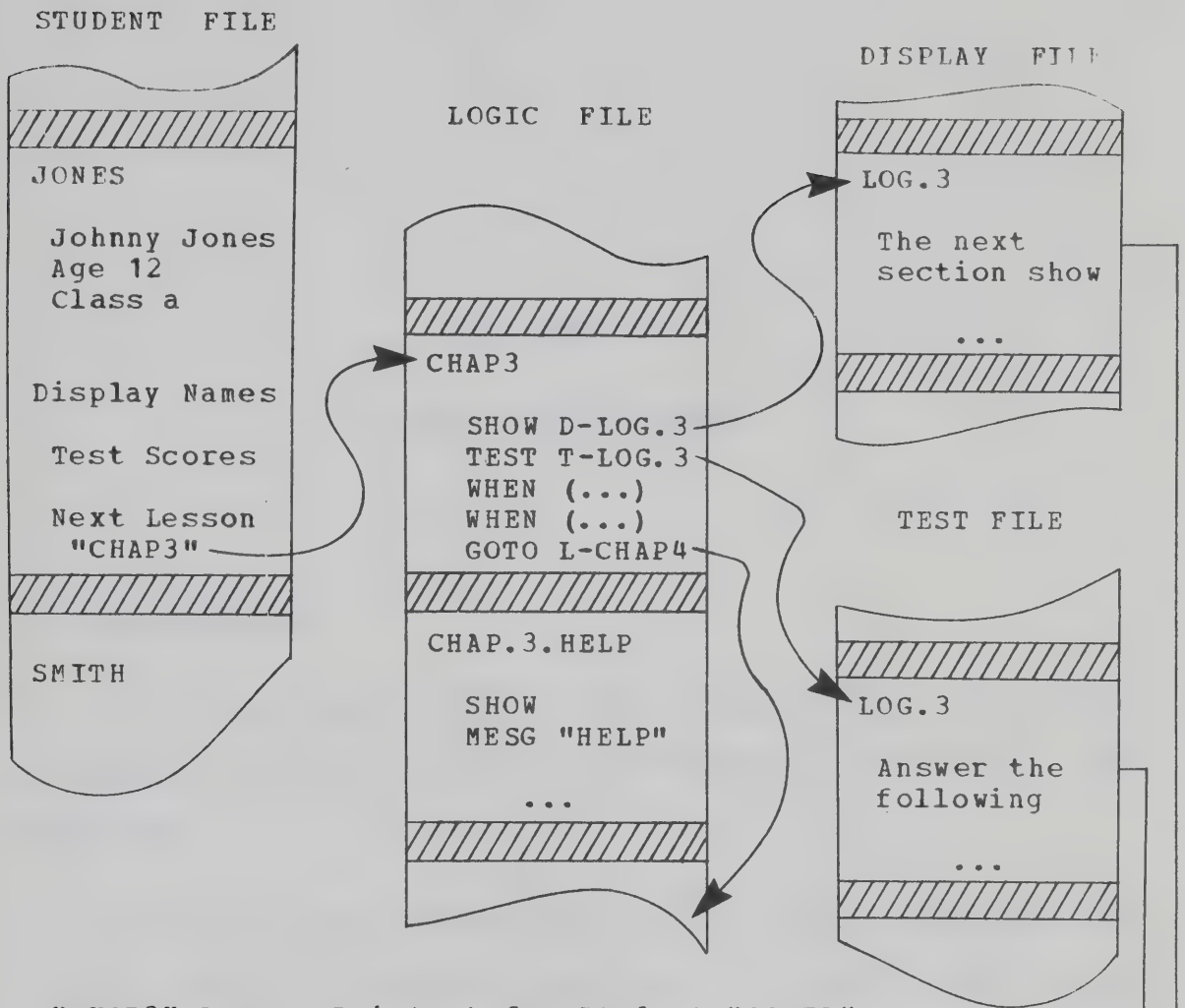
2.5 Day (Logic) Construction

The reader is referred to APPENDIX B for a syntactic description of Day constructs.

Type	#	Sample Statement
Lesson Logic Statements	1	SHOW D-LOG.3
	2	TEST T-LOG.3
Decision Logic Statements	3	WHEN (T-LOG.3 GE 80%) GOTO L-ENRICHMENT
	4	WHEN (T-LOG.3 LT 50%) GOTO L-CHAP.3.HELP
	5	GOTO L-CHAP4

Figure 2.5 Typical Day Construct

Figure 2.5 shows how Logic Statements might be arranged within a Day named "CHAP3". Statements numbered 1 and 2 define the lesson to be printed out for a student, while statements 3, 4 and 5 define the decisions (and their parameters) to be used in determining the student's events for the next day. Figure 2.6 shows how the Lesson Logic Statements of the Day named "CHAP3" interact with the various TAIM files in order to produce an individual's lesson.



"CHAP3" Lesson Printout for Student "JONES"

.	JONES	Johnny Jones	Class-A	.
.				.
.	DAY : CHAP3	DATE : xx/xx/xx		.
.				.
.	DISPLAY : LOG.3			.
.	The next section shows how to add numbers using logarithms
.				.
.	TEST : LOG.3			.
.	Answer the following questions on your mark sense cards
.				.
.				.

Figure 2.6 Using the Student, Logic, Display and Test Files to Produce a Typical Student Lesson

CHAPTER III

TAIM - SYSTEM DESIGN

3.1 Introduction

The TAIM System (Figure 3.1), consists of two major subsystems: an Interactive TAIM Subsystem and a Batch TAIM Subsystem.

The Interactive TAIM Subsystem is non-scheduled, that is, it may be activated any number of times a day, at any hour, by any registered user. Under authorized user control, this Subsystem will: (1) initialize TAIM system files; (2) register and unregister both teachers (users) and students; (3) monitor student progress; (4) retrieve student data; (5) mark and record student question responses; (6) alter the sequence and rate of instruction for individual or groups of students; and (7) create, destroy or modify Days, Displays and Tests.

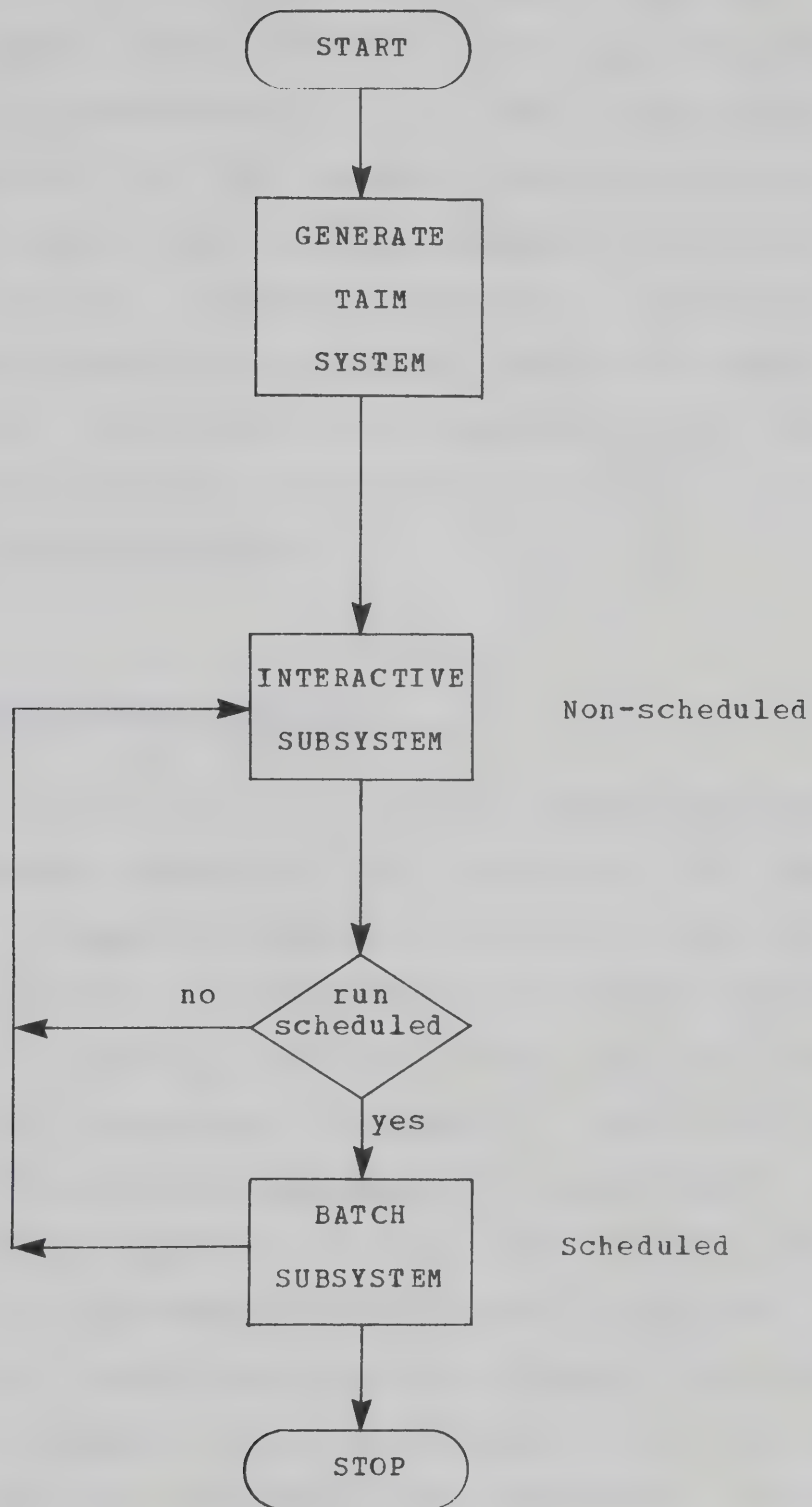


Figure 3.1 The TAIM System

The Batch TAIM Subsystem is scheduled, that is, it is activated once a night (before a school day) by a computer operator at the data center. It is the responsibility of this Subsystem to: (1) produce lesson printouts for each registered student; (2) print any messages directed to the teacher during Logic Statement interpretation; (3) accumulate statistics on system usage; (4) execute any TAIM commands which have been placed in a batch queue for overnight (over weekend) processing; and (5) perform any required system maintenance.

3.2 Interactive TAIM Subsystem

From Figure 3.2 it is readily observed that users at remote terminals communicate with the system via commands. Entering a command establishes a scope of work. For some commands, the scope of work encompasses several operations that can be separately identified. After entering a command, one of the separately identifiable operations may be activated by entering a subcommand. A subcommand, like a command, is a request for work also, however, the work requested by a subcommand is a particular operation within the scope of work established by a command. The reader is referred to APPENDIX A for a description of the TAIM command language, that is, the commands and subcommands recognized by the Command Monitor.

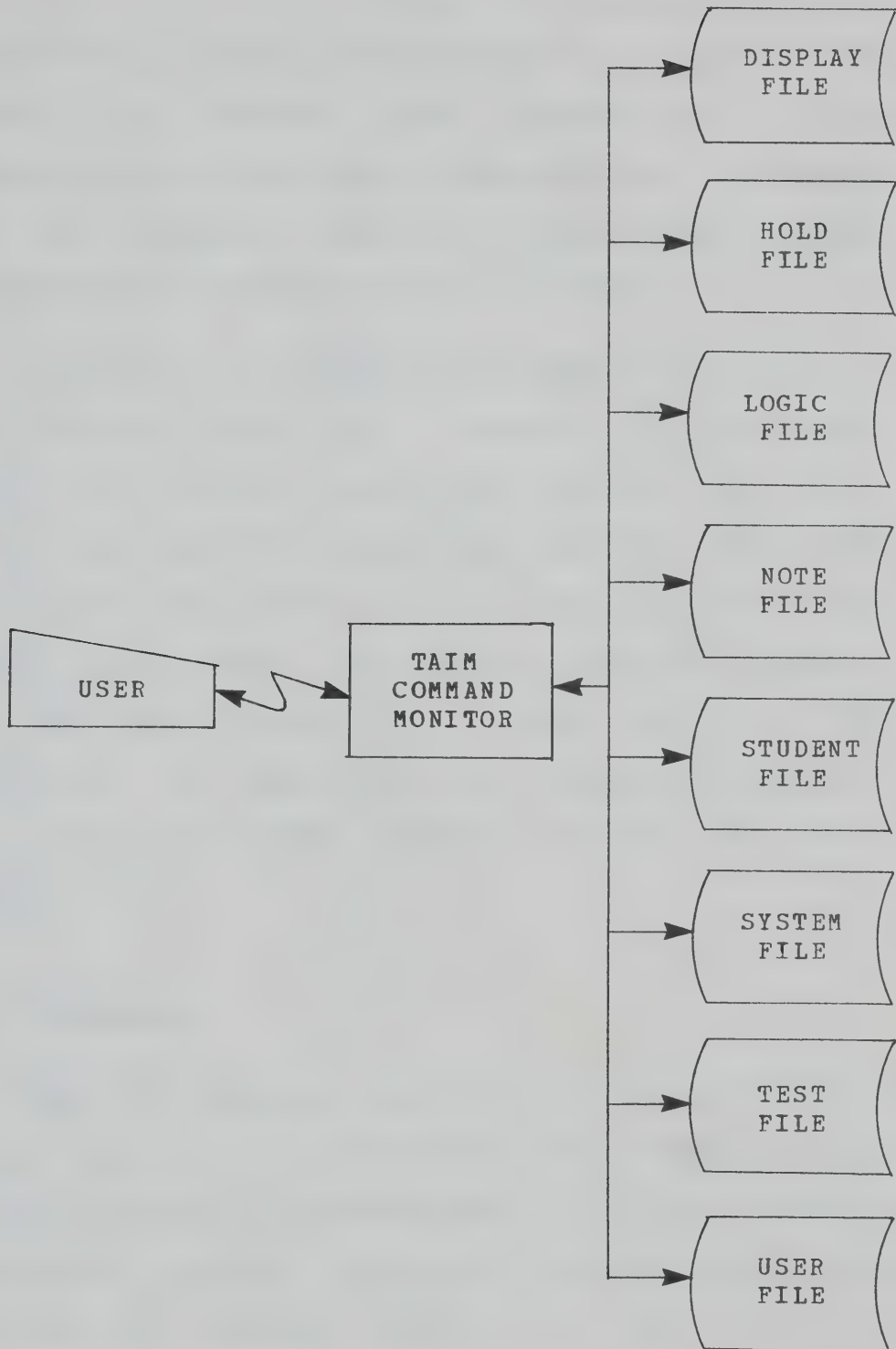


Figure 3.2 Interactive TAIM Subsystem

There are three levels of responsibility in TAIM; consequently, users are classified as either Teachers, Authors or Proctors; and commands are respectively categorized as either Mode 1 Commands, Mode 2 Commands or Mode 3 Commands. Table 3.1 lists TAIM commands and subcommands in alphabetical order by mode.

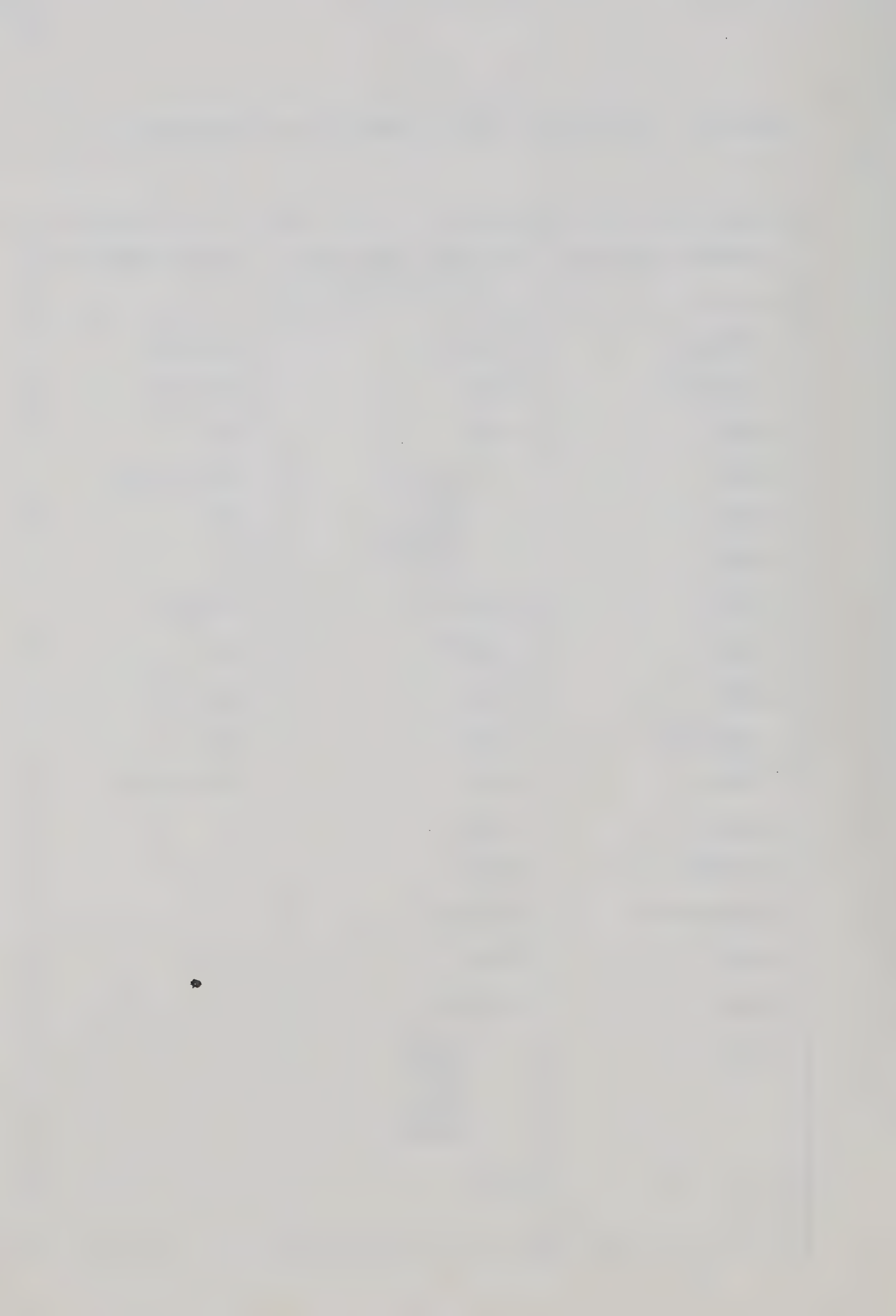
Proctors are managers of the TAIM System. They register and unregister users, and in general are responsible for keeping the system operational. Authors are users who control the nature of instruction available from TAIM and are therefore responsible for creating and maintaining Days, Displays and Tests. Teachers are classroom teachers using TAIM for the instruction of their students. They are responsible for registering and unregistering students and for monitoring student progress through the curriculum network.

Mode 1 Commands

Mode 1 commands, accessible by all registered users, may be used to: (1) register and unregister students; (2) modify student registrations; (3) retrieve student data; (4) monitor student progress; (5) enter student question responses; (6) examine and/or alter next Day pointers; (7) suspend student lesson production for a number of days; and (8) direct notices to individual(s) or groups of students.

Table 3.1 Summary of TAIM Commands and Subcommands

Command (Mode 1)	Command (Mode 2) Subcommand	Command (Mode 3)
AVERAGE	CATALOG	CONDENSE
DISPLAY	DROP	DISPLAY
HOLD	EDIT	DUMP
MARK	Delete	INITIALIZE
MODE	Insert	MODE
NOTE	Print	MTS
	Replace	
	Stop	
POINT	EVALUATE	OFFLINE
QUIT	EXECUTE	QUIT
READ	FIND	REGISTER
REGISTER	LABEL	SET
TRACE	MODE	UNREGISTER
UNHOLD	PLACE	WHY
UNNOTE	QUIT	
UNREGISTER	RENAME	
WHEN	SAVE	
WHO	SETKEY	
WHY	Clear	
	Print	
	Set	
	Stop	
	Weight	
	WHY	



Mode 2 Commands

Using Mode 2 Commands, an Author or Proctor may:

- (1) list the names of all current Days, Displays and Tests;
- (2) place a copy of any Day, Display or Test into one of three available workspaces;
- (3) define a new Day, Display or Test in any of the workspaces;
- (4) edit the contents of any workspace;
- (5) assign a label to any of the workspaces;
- (6) save (copy) the contents of any workspace into either the Display File, Logic File or Test File;
- (7) have listed all those Days whose Logic Statements make reference to a specified Day, Display or Test; and
- (8) delete any current Day, Display or Test from the system files provided that it is not referenced by any Logic Statement in any Day.

Mode 3 Commands

Using Mode 3 Commands, a proctor may: (1) initialize or condense (repack) TAIM files; (2) display the status of TAIM commands and registered users; (3) register and unregister TAIM users; (4) place TAIM commands into a queue for overnight batch processing; (5) alter the status of TAIM commands and registered users; and (6) interact directly with the operating system under which TAIM is being run.

The full set of TAIM commands is not accessible by all registered users. Figure 3.3 shows the responsibility levels of the three user groups and the commands available to each.

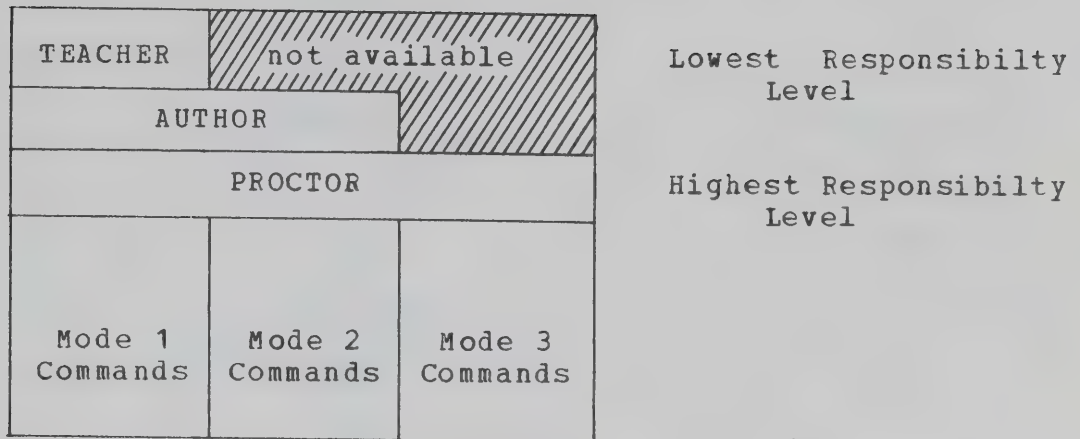


Figure 3.3 User Responsibility Levels

Command Monitor

When the Interactive TAIM Subsystem is invoked the Command Monitor (Figure 3.4) assumes control and subsequently prompts the user for an identification. Upon receiving an acceptable identification, the Monitor initializes the currently active mode according to Table 3.2, and informs the user that it is "READY" to accept a command.

When user is a	Currently active mode becomes
Teacher	Mode 1
Author	Mode 2
Proctor	Mode 3

Table 3.2 Initializing the Currently Active Mode

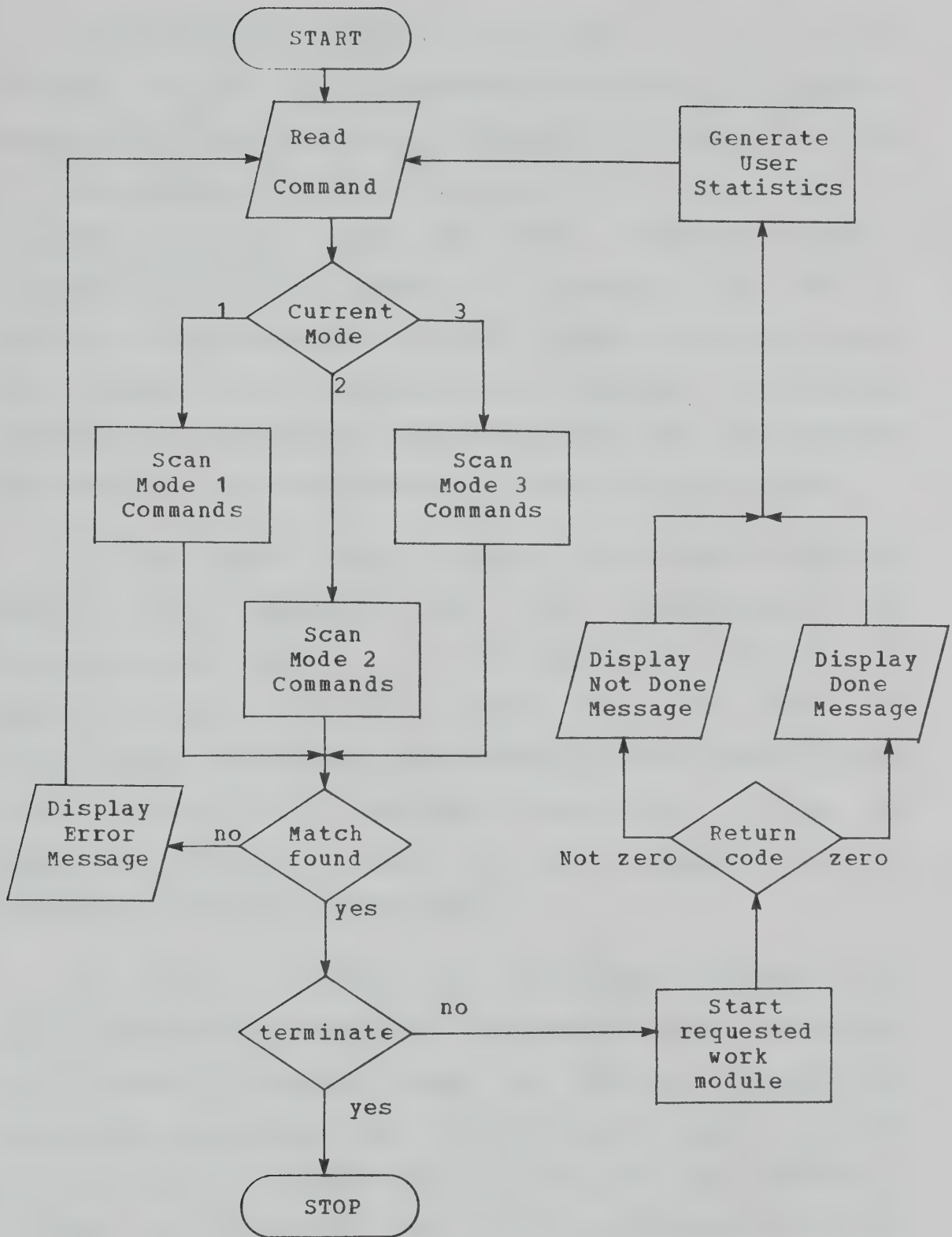


Figure 3.4 Command Monitor

A user-entered command is accepted if the command belongs to the set of commands defined for the currently active mode; otherwise, it is rejected. For example, if Mode 2 is currently active then the Monitor will accept Mode 2 Commands and will reject any Mode 1 Commands or Mode 3 Commands. If an input command is rejected, the user is notified and the Monitor is again "READY" to receive input. On the other hand, if a command is accepted, the Monitor initiates the appropriate work module (one for each command) and waits for the work module to complete its processing.

A work module uses a numeric return code to indicate whether the requested work was successfully or unsuccessfully completed. A zero code means that the work was accomplished, whereas, a nonzero code means that for some reason (the reason is reflected in the numeric return code) the work was not performed. Depending on whether the return code is zero or nonzero the user is informed that the command was "DONE" or "NOT DONE".

In order to obtain data on command utilization and costs involved in running the Interactive TAIM Subsystem, each command accepted causes the Monitor to record the following statistical data: (1) the user's name; (2) the date and time of command entry; (3) the CPU time required to perform the requested work; (4) the approximate cost of performing the work; (5) a numeric code indicating whether the command was successfully or unsuccessfully completed;

and (6) the command as actually entered by the user.

Following this, the Command Monitor is once again "READY" to receive user input.

Attention Interruption

The attention interruption allows the user to interrupt the processing of any command which intentionally or inadvertantly results in a great volume of output being displayed at the terminal. When the attention key is used to interrupt a command the Monitor queries "DO YOU WANT THIS COMMAND BATCHED? - YES OR NO". If the response is "NO" the Monitor simply goes into the "READY" state. However, if the response is "YES" the Monitor places the input command into a specially provided queue for overnight batch processing before going into the "READY" state.

Decision Logic Statement Interpretation

One of the more important work modules in the interactive TAIM Subsystem (see Figure 3.5) is one which reads a student's response card; marks any question responses contained therein according to an answer key and stores the results in the student's diary; locates the student's current Day in the Logic File and interprets the Decision Logic Statements, thereby determining which Day in the curriculum network the student is to receive next.

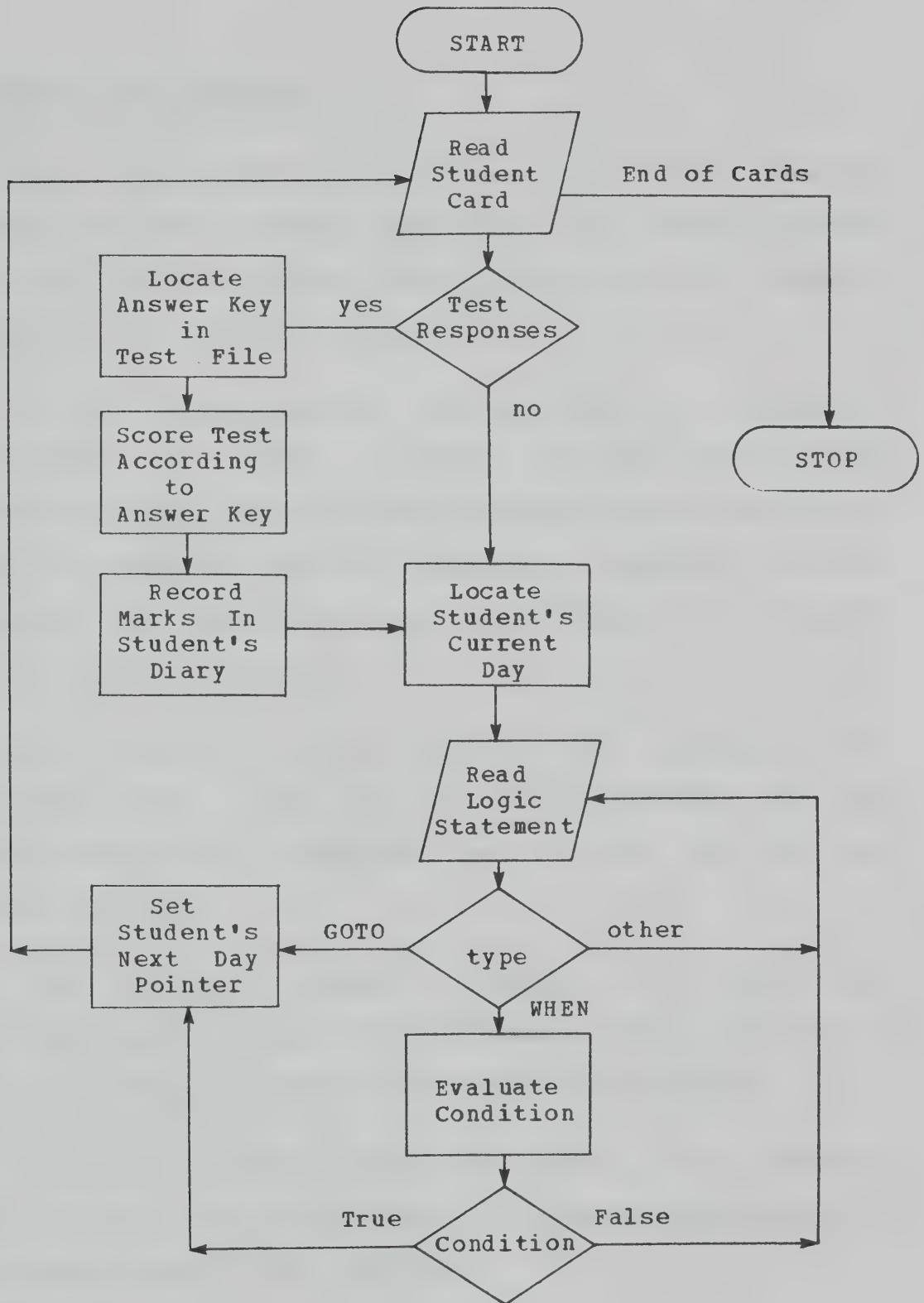


Figure 3.5 Decision Logic Statement Interpretation

3.3 Batch TAIM Subsystem

The main components of the Batch TAIM Subsystem (Figure 3.6) are the Batch Processor, the Message Printer and the Command Monitor. Invoking the Batch TAIM Subsystem results in the following events:

(1) The Batch Processor assumes control, accumulates statistical data onto a history tape and produces three discrete outputs, each of which requires further processing. They are unsorted teacher messages, unsorted student printouts and unsorted TAIM commands destined for overnight Command Monitor execution.

(2) Unsorted teacher messages are arranged into ascending order first by class then by message type. The sorted messages are summarized and printed out via the Message Printer.

(3) Unsorted student printouts are sorted into ascending order by class then by student and are printed out directly using an exit from the system sort routine.

(4) Unsorted TAIM commands are sorted into ascending order by user. Upon being sorted the commands are passed to the Command Monitor for execution.

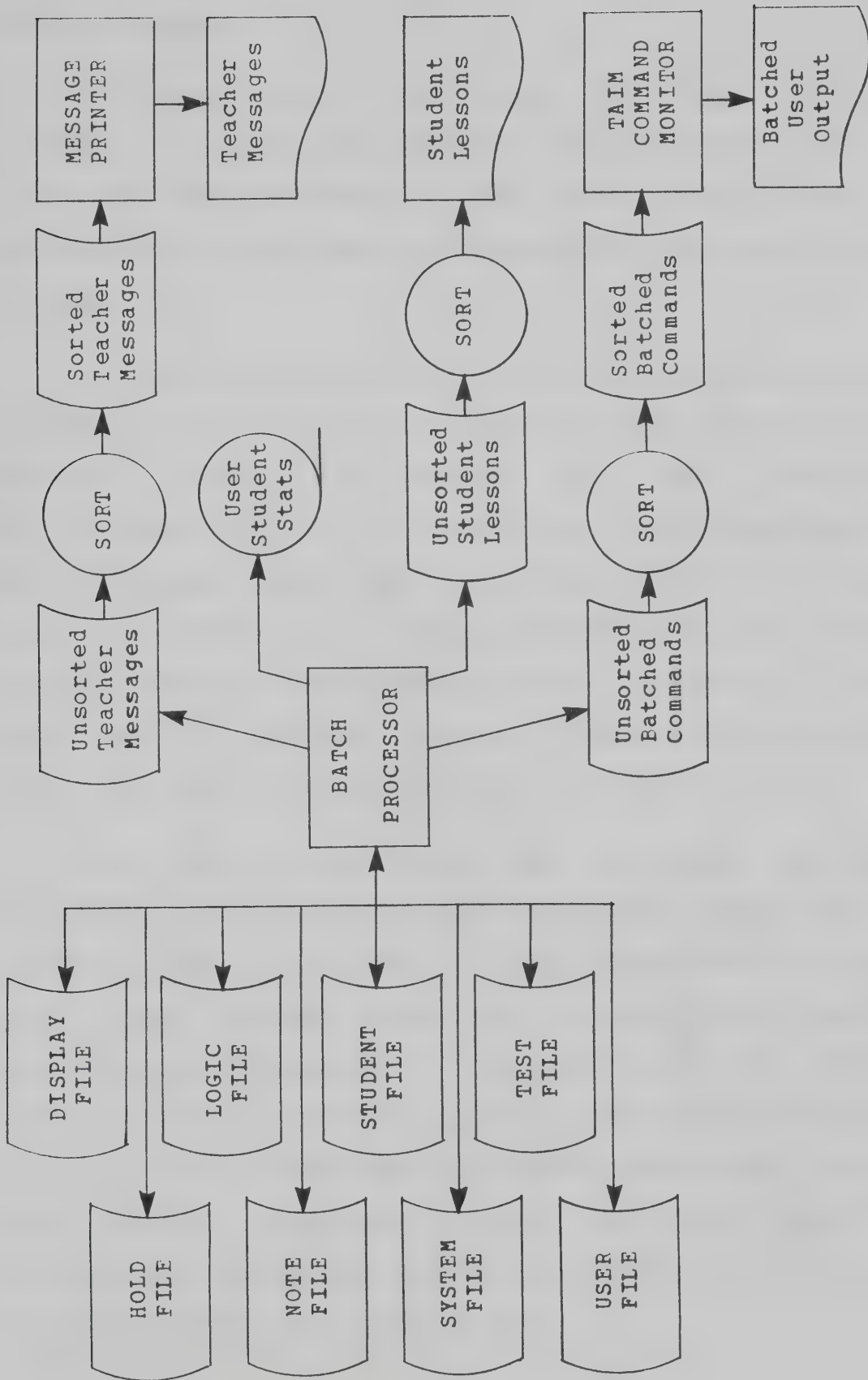


Figure 3.6 Batch TAIM Subsystem

Batch Processor

For each student registered, the Batch Processor (Figure 3.7) prints the student's identification at the top of a new page followed by the name of the last Day encountered and the question responses for any Test(s) taken in that Day.

If the student's lesson is to be suppressed, both the student and the teacher are informed of this action, and no further output is produced for that individual. On the other hand, if the lesson is not to be suppressed the Batch Processor prints any notices which have been queued for the student and records the notices in the student's diary; retrieves the student's next Day indicator; locates that Day in the Logic File and by interpreting the Lesson Logic Statements therein produces the student's lesson.

When all students have been processed, the Batch Processor takes all user/student statistics which have been gathered since the last batch run and accumulates them on a long term storage medium for retrospective analysis; recovers any TAIM commands in the Batch queue and prepares them for Command Monitor execution; empties the statistics queue, the note queues and the command batch queue; updates all current suspension counts (see HOLD command in APPENDIX A); and informs the teacher of those students whose suspension counts have reached zero.

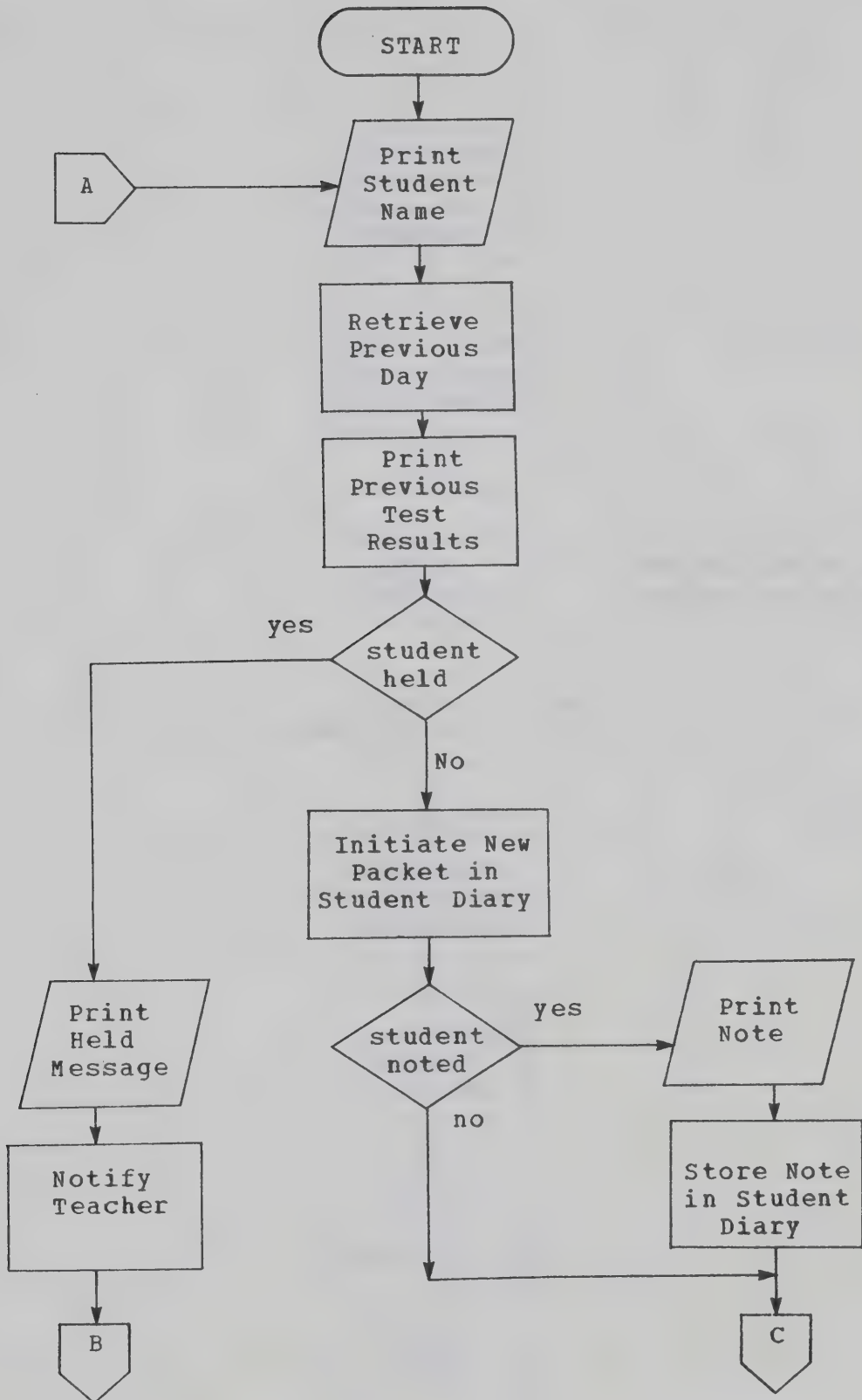


Figure 3.7 Batch Processor (continued on next page)

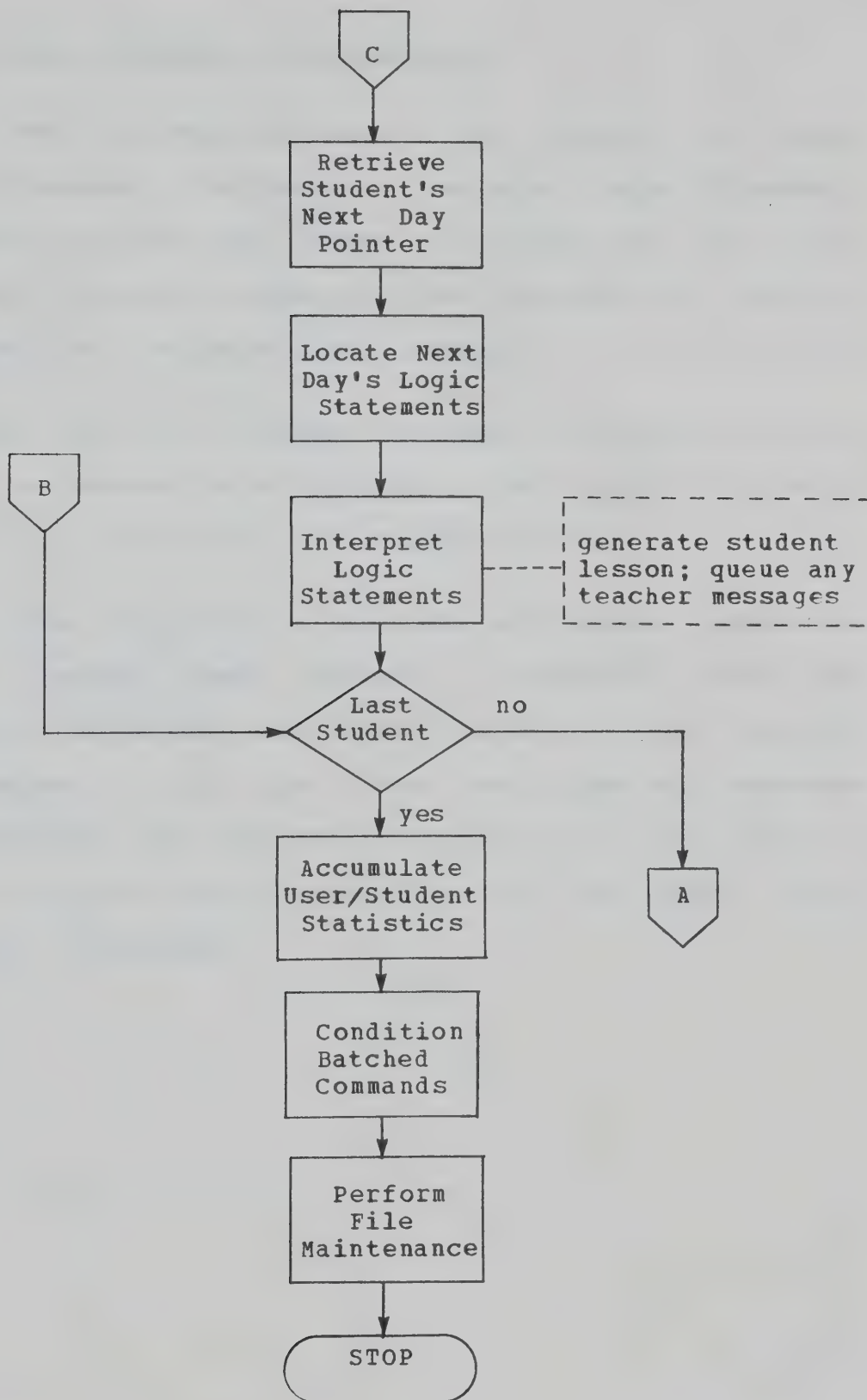


Figure 3.7 Batch Processor (continued from previous page)

Lesson Logic Statement Interpretation

As was discussed earlier, a Day consists of Lesson Logic Statements followed by Decision Logic Statements. Interpretation of a Day (Figure 3.8) begins with the first statement, proceeds sequentially and terminates at the first occurrence of a Decision Logic Statement.

Each time that a student is given a Display or Test, or causes a Message to be directed to the teacher a record of the fact is entered into the student's diary.

In order to determine the costs involved in generating student lessons, each statement interpreted causes the following statistical data to be recorded: (1) the name of the student; (2) the date and time of statement interpretation; (3) the CPU time required; (4) the number of lines of printer output produced; and (5) the type of Logic Statement interpreted.

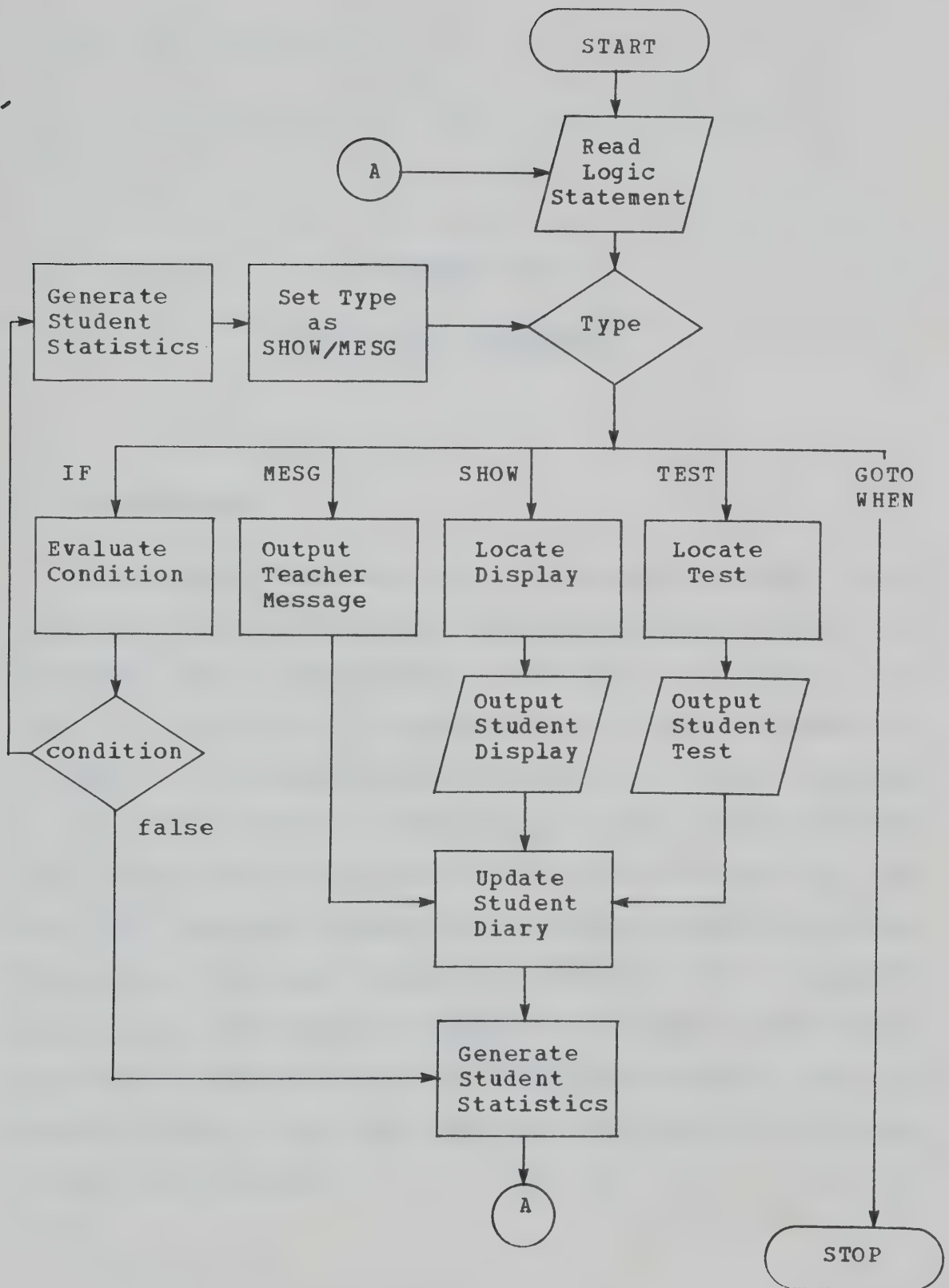


Figure 3.8 Lesson Logic Statement Interpretation

CHAPTER IV

TAIM - FILE STRUCTURES

4.1 Introduction

The primary importance of a programmed system which involves the manipulation of large quantities of data is an efficient file organization technique. A "file" is an organized collection of "records" and in turn, a "record" is a number of interrelated data elements. A "record" consists of two parts: a "key" - used to distinguish between records; and a "function" - that part of the "record" which is not the key. Although there are various methods of handling files, the technique ultimately adopted for a specific application is normally dependent on criteria such as the importance of fast access, the importance of being able to quickly update the file and the importance of optimizing storage requirements.

4.2 The TAIM System Files

Two requirments of the TAIM System are that data be available to online users and that response times be minimal. Consequently, TAIM file structures are designed for use with direct (random) access devices. File structures have been developed for each of the following TAIM files:

- (1) System File,
- (2) Display File,
- (3) Hold File,
- (4) Logic File,
- (5) Note File,
- (6) Student File,
- (7) Test File,
- (8) User File.

In total there are thirty-three unique record types utilized in configuring the above files. However, none of the files requires the use of all record types and no two files make use of the same subset of records. Table 4.1 lists the various record types in alphabetical order, indicates to which file(s) each belongs and gives the page (APPENDIX C) on which a record layout for each record type may be found. Each file will be discussed in the order noted above.

Table 4.1 Record-File Associations

FILE RECORD	DISPLAY	HOLD	LOGIC	NOTE	STUDENT	SYSTEM	TEST	USER	layout on page
BATCH				X					143
COMMAND SECTION						X			134
DISPLAY SECTION						X			134
ENCODED LOGIC			X						142
ERROR SECTION						X			136
FILE HEADER	X		X		X		X	X	138
HOLD		X							144
HOLD INFORMATION		X							144
HOLD SECTION						X			136
INDEX	X		X		X		X	X	138
LOGIC SECTION						X			134
NOTE				X					143
NOTE INFORMATION				X					143
NOTE SECTION						X			136
OFFLINE SECTION						X			137
ONLINE SECTION						X			137
REFERENCE	X		X				X		141
SECTION HEADER						X			133
STATISTICS								X	144
STATISTICS HEADER								X	144
STUDENT DISPLAY					X				140
STUDENT LEADER					X				139
STUDENT MESSAGE					X				140
STUDENT NOTE					X				140
STUDENT PACKET					X				139
STUDENT SECTION						X			135
STUDENT TEST					X				140
SYSTEM HEADER						X			133
TFS1-KEY							X		141
TEST SECTION						X			135
TEXT	X						X		141
UNIT LEADER	X		X				X		141
USER SECTION						X			135

System File

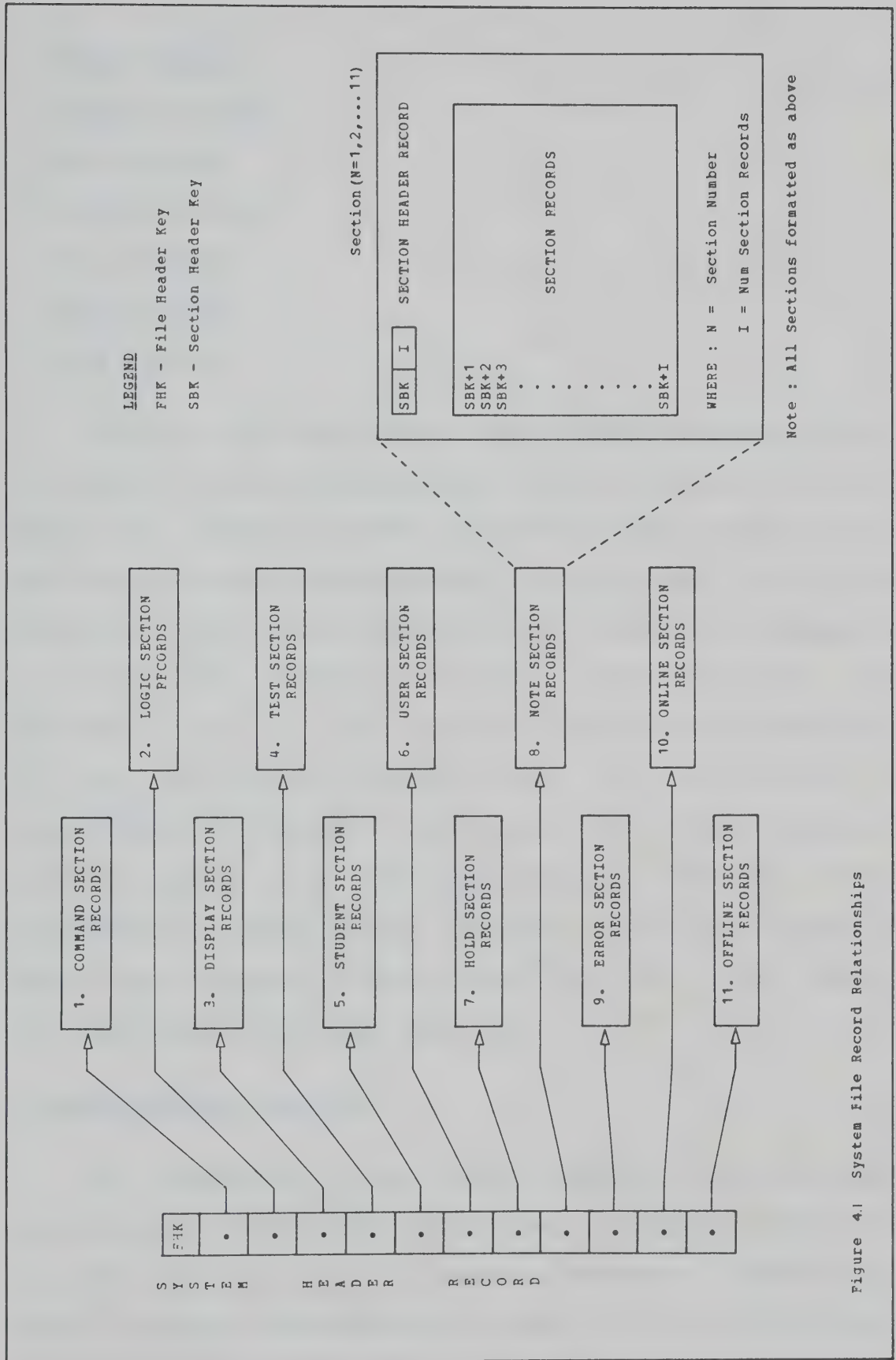
The System File (Figure 4.1) contains information central or essential (kernel) to the operation of the TAIM System. Within this file is retained the TAIM command table, command error messages, lists of program-dependent logical I/O assignments and initial record configurations for each of the other TAIM files.

Before the TAIM System can be utilized, the System File must be pre-loaded with kernel information by a knowledgeable TAIM programmer. Refer to APPENDIX D for a sample System File. Subsequently, the programmer invokes the Interactive TAIM Subsystem, initializes all other TAIM files, registers users (proctors, authors, teachers) and in general prepares the System for global use.

The System Header Record contains eleven pointers, one to each of eleven groups of records known as "Sections". Each Section, containing a Section Header Record followed by a specified number of Section Records, is described as follows:

1. Command Section

This Section contains the TAIM command table. Each Command Section Record defines a command - its name, the number of characters accepted as its abbreviation, the mode to which it is to be associated, its current status (active or inactive) and its assigned command number.



LEGEND

FK - File Header Key

SBK - Section Header Key

2. Logic Section
3. Display Section
4. Test Section
5. Student Section
6. User Section
7. Hold Section
8. Note Section

Records contained within each of these Sections define the initial record configuration of the corresponding TAIM file. For example, records within the Logic Section define the initial record configuration of the Logic File. The "function" part of any Section Record contains the image of a corresponding TAIM file record. For example, the "function" part of a Test Section Record contains the image of a Test File Record. Initialization of a TAIM file is accomplished by simply transcribing the images contained within a Section into the appropriate locations of the corresponding file. For example, the Note File is initialized by moving record images from the Note Section into positions within the Note File.

9. Error Message Section

The execution of any TAIM command terminates with either a zero return code (indicating successful completion) or a nonzero return code (indicating abnormal termination). Error messages corresponding to each of the nonzero return

codes are located simply by taking the Error Message Section Base Key and adding to it the assigned command number and the nonzero return code.

10. Online Section

A logical I/O unit is a symbolic name which is used in programs to specify the source of data for input or the destination of output information. A logical I/O unit does not name a specific physical file or device; it simply serves as a reference. When a program is run, it becomes necessary to specify for each logical I/O unit used, which physical file or device should be used. Thus, this section contains a list of all the Command Monitor's logical I/O units and the physical files to which each is to be attached.

11. Offline Section

This section contains a list of all the Batch Processor's logical I/O units and their associated physical files.

Display File

The Display File (Figure 4.2) consists of a File Header Record, an Index or Directory and a series of record groupings called "Display Units". There is an Index Record and Display Unit corresponding to each Display in the TAIM

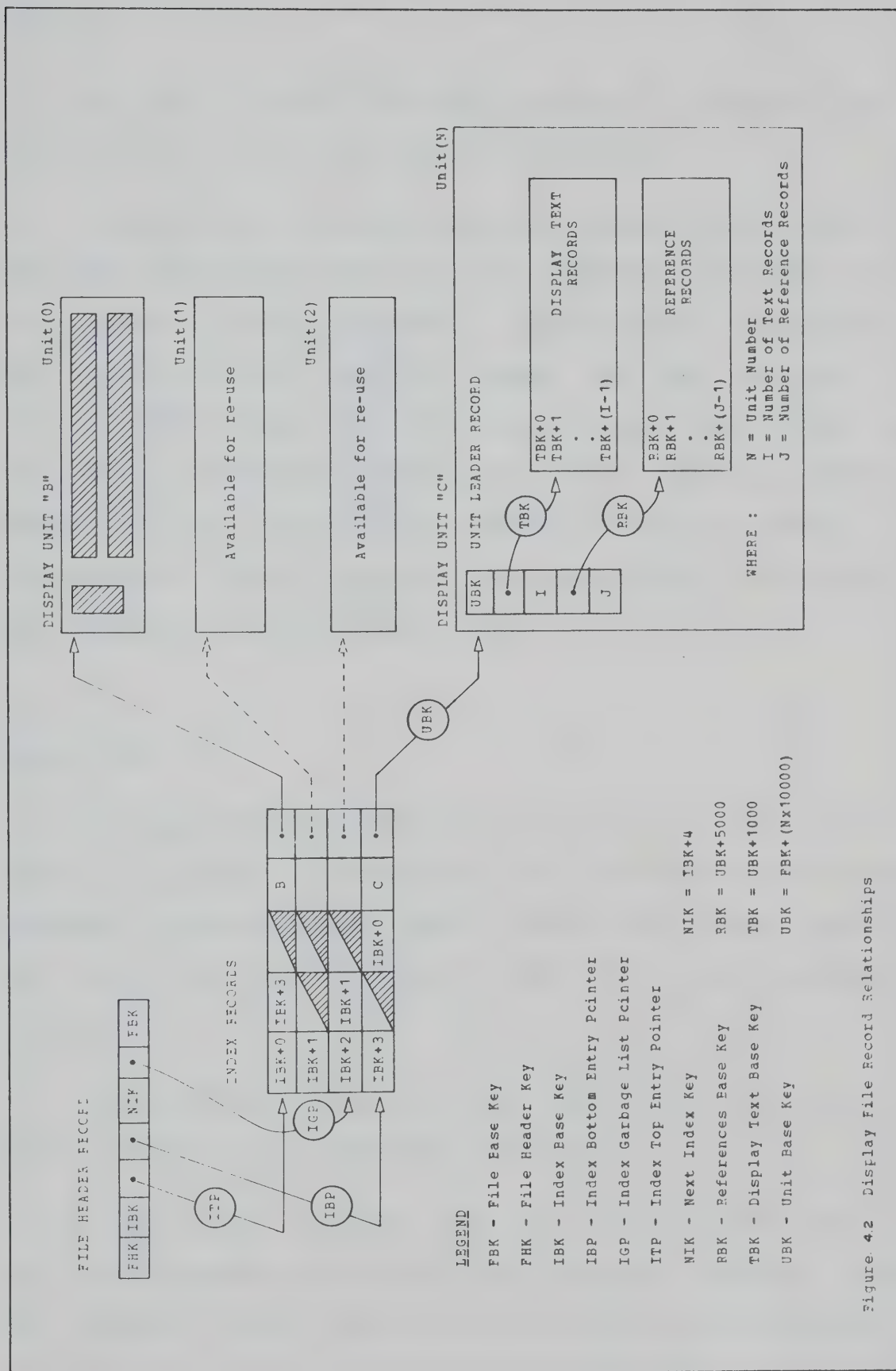


Figure 4.2 Display File Record Relationships

System.

An Index Record contains the name of a Display and a pointer (UBK) to its associated Display Unit.

A Display Unit consists of a Unit Leader Record, one or more Text Records and zero or more Reference Records. The Unit Leader Record contains a pointer (TBK) to the first Text Record, a count of the number of Text Records, a pointer (RBK) to the first Reference Record, and a count of the number of Reference Records. The Text Records contain Display text and the Reference Records contain the names, in alphabetical order, of those Days whose Logic Statement(s) make reference to this Display Unit.

Logic File

The Logic File (Figure 4.3) consists of a File Header Record, an Index or Directory, and a collection of record groupings known as "Logic Units". There is an Index Record and Logic Unit corresponding to each Day in the curriculum network.

An Index Record contains the name of a Day and a pointer (UBK) to its associated Logic Unit.

A Logic Unit consists of a Unit Leader Record, one or more Logic Records and zero or more Reference Records. The Unit Leader Record contains a pointer (LBK) to the first

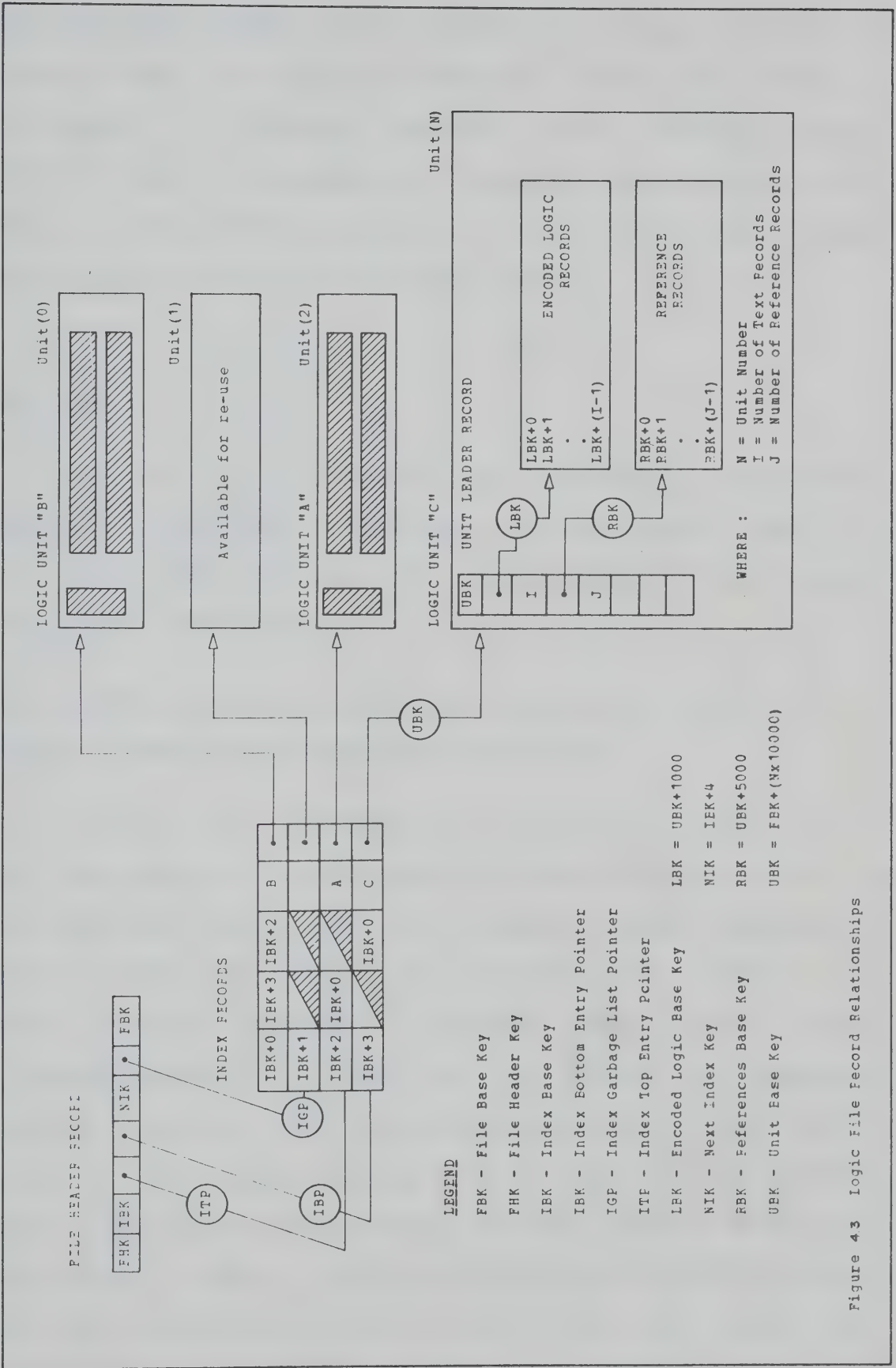


Figure 43 Logic File Record Relationships

Logic Record, a count of the number of Logic Records, a pointer (RBK) to the First Reference Record and a count of the number of Reference Records. Logic Records contain encoded Logic Statements and Reference Records contain the names, in alphabetical order, of those Days whose Logic Statement(s) reference this Logic Unit.

Test File

The Test File (Figure 4.4) contains a File Header Record, an Index and a collection of "Test Units". There is an Index Record and Test Unit corresponding to each current Test in the TAIM System.

An Index Record contains the name of a Test and a pointer (UBK) to its associated Test Unit.

A Test Unit consists of a Unit Leader Record, one or more Text Records, zero or more Reference Records and one or more Test-Key Records. The Unit Leader Record contains a pointer (TBK) to the first Text Record, a count of the number of Text Records, a pointer (RBK) to the first Reference Record, a count of the number of Reference Records, a pointer (KBK) to the first Test-Key Record and a count of the number of Test-Key Records. The Text Records contain Test text, the Reference Records contain names, in alphabetical order, of Days referencing this Test Unit and the Test-Key Record contains a test weight and answer key

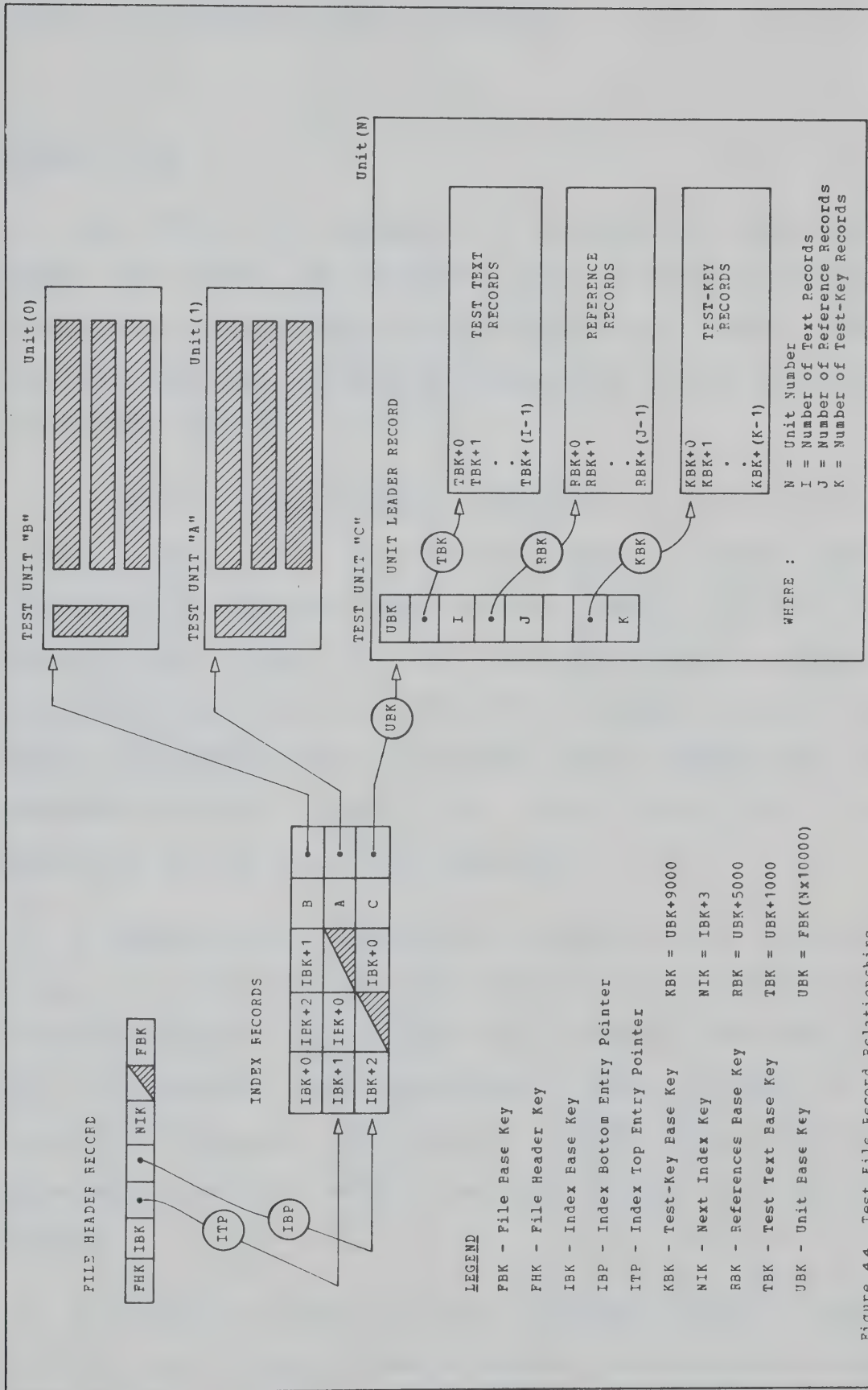


Figure 44 Test File Record Relationships

for use in marking student responses.

Student File

The Student File (Figure 4.5) consists of a File Header Record, an Index or Directory and a collection of record groups known as "Student Units". There is an Index Record and corresponding Student Unit for each student registered in the TAIM System.

An Index Record contains a student's identification, a pointer (PFK) to his Student Unit, a class letter (A,B,...Z), a Hold File Entry Pointer (HFP), a Note File Entry Pointer (NFP) and an attendance-flag. A student's attendance-flag is turned "on" when a response card for the student is entered into the system via the Interactive TAIM Subsystem and is turned "off" again during the next invocation of the Batch TAIM Subsystem.

A Student Unit consists of a Student Leader Record and a chain of linked (forward and back) record groupings called "Student Packets". There is one such Packet for each Day that the student has encountered (been through) in the curriculum network. The Student Leader Record contains registration data as well as pointers to his first and last Packets.

Each Student Packet contains a Student Packet Record, zero or more Student Message Records, zero or more Student

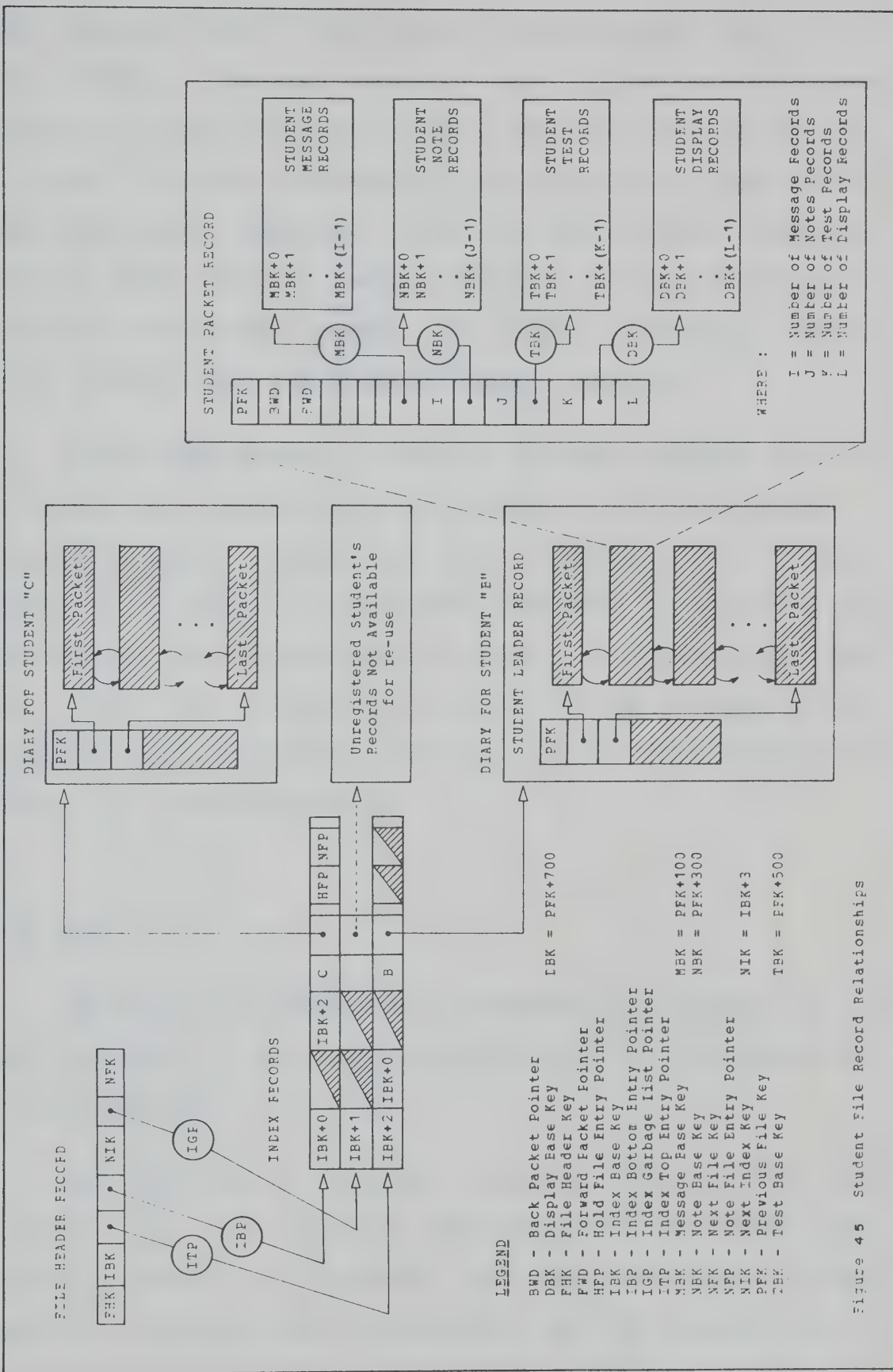


Figure 45 Student File Record Relationships

Note Records, zero or more Student Test Records and zero or more Student Display Records. The Student Packet Record contains forward and back Packet links; the name of the Day to which a Packet corresponds; the name of the next Day in the chain; base keys for each of the Student Message, Student Note, Student Test and Student Display Records; and counts of the current number of Student Message, Student Note, Student Test and Student Display Records.

A new last packet is created for each student whenever the Batch TAIM Subsystem is activated. In it is placed a Student Message Record for each message that a student causes to be sent to the teacher; a Student Note Record for each notice that he receives; a Student Test Record for each Test that the student is to take; and one or more Student Display Records containing the names of all Displays printed out in the student's lesson.

Hold File

The Hold File (Figure 4.6) consists of a fixed set of Hold Records, a Hold Information Record and a variable set of Hold Records.

The fixed set of records contains twenty-seven Hold Records, one for each of the student hierarchies (ALL, CLASS-A, CLASS-B, ... CLASS-Z). The Hold Information Record contains a pointer (HBK) to the start of the variable set of

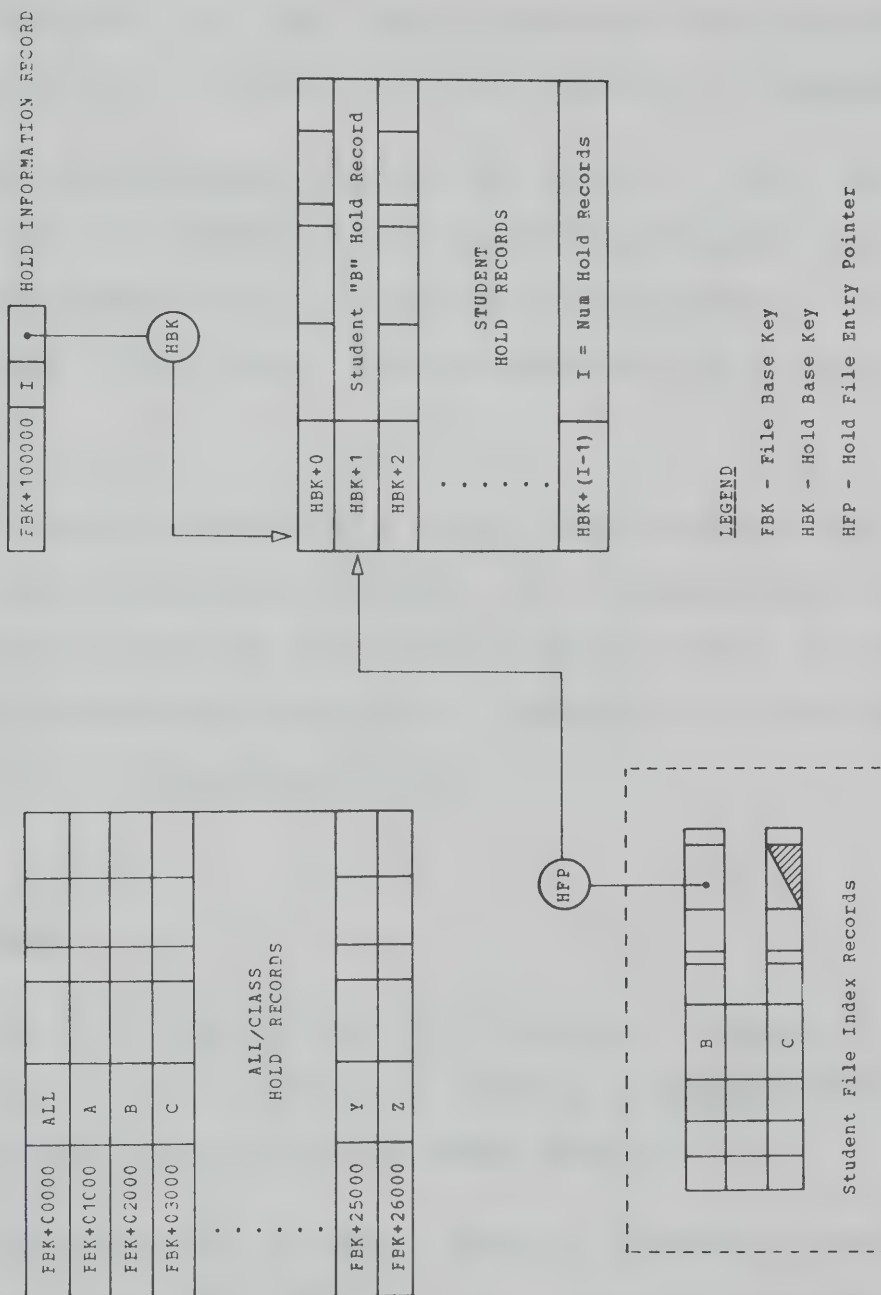


Figure 4.6 Hold File Record Relationships

Hold Records and a count specifying the current number of records in that set. A Hold Record contains information such as the identification of the student being held; the identification of the user initiating the suspension; the date held and the number of school days to be suspended.

If an individual student is held, a Hold Record is allocated and placed at the end of the variable set of Hold Records. Subsequently the key of this record is stored in the Hold File Entry Pointer (HFP) of the student's Index Record.

If ALL/CLASS is held, rather than create a Hold Record for each registered student, the corresponding ALL/CLASS Hold Record from the fixed set of Hold Records is recovered and the status-flag therein is turned "on" to indicate that ALL/CLASS is under suspension.

Note File

The Note File (Figure 4.7) contains a fixed set of Note Records, a Note Information Record, a variable set of Note Records and a collection of Batch Records.

The fixed set of Note Records contains twenty-seven records, one for each of the student hierarchies (ALL, CLASS-A, CLASS-B, ... CLASS-Z). The Note Information Record contains a pointer (NBK) to the first Note Record in the variable set, a count of the number of records currently in

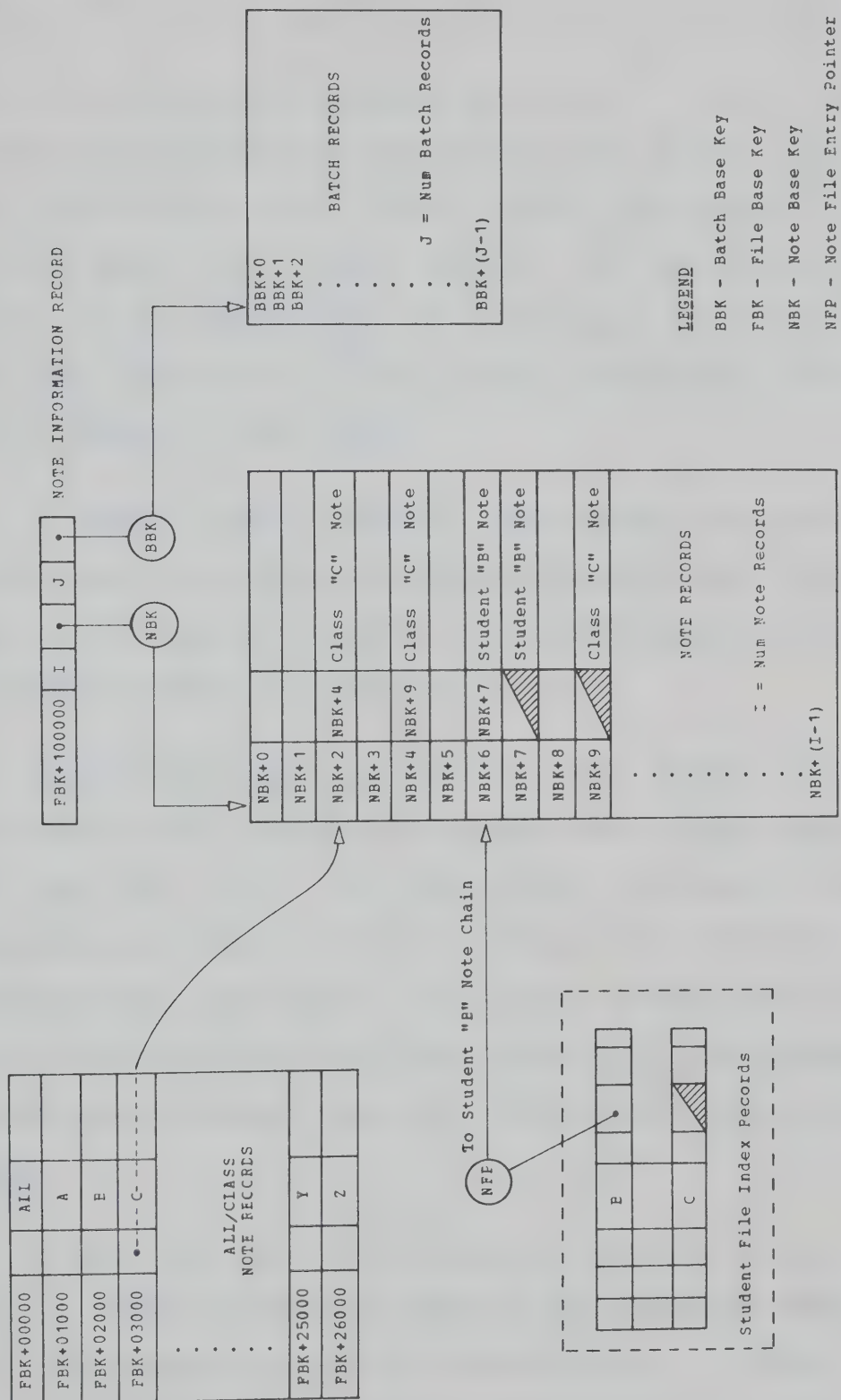


Figure 4.7 Note File Record Relationships

the variable set, a pointer (BBK) to the first Batch Record and a count of the number of current Batch Records.

If an individual student is noted, a Note Record is created and appended to the variable set of Note Records. If this is his first note, the key of the Note Record is stored in the Note File Entry Pointer of the student's Index Record. On the other hand, if this is not his first notice, the newly created Note Record simply becomes the last record of the student's note chain.

A Note Record contains a link to the next Note Record in the chain (queue), the identification of the student to whom the notice is intended, the identification of the user sending the note and the note itself.

If ALL/CLASS is to be noted, a Note Record is created and added to the end of the variable set of Note Records. If this is the first ALL/CLASS note, the key of the newly created Note Record is stored in the Note Pointer of the corresponding ALL/CLASS Note Record. On the other hand, if this is not the first ALL/CLASS note, the newly created Note Record simply becomes the last record in the ALL/CLASS note chain.

A user may, via the Interactive TAIM Subsystem, request that a TAIM command be retained for overnight processing. Each such request causes the generation of a Batch Record which contains the requesting user's identification and the

command whose execution is to be deferred.

User File

The User File (Figure 4.8) contains a File Header Record, an Index, a Statistics Header Record and zero or more Statistics Records. To each registered user belongs an Index Record containing the following information: the user's name and identification; the user's status (teacher, author, proctor); date of registration; date of first and last use of the Interactive TAIM Subsystem; total number of uses since being registered and a batch-flag which is turned "on" when the user places a TAIM command in the batch queue for overnight processing.

As a user converses with the Interactive TAIM Subsystem, each accepted command causes the generation of a new Statistics Record. Each such record contains the following information: the input command and its assigned command number; the user's identification; the date and time of entry; the CPU time required to execute the command; the approximate cost of executing the command; and a numeric return code indicating whether the command was successfully or unsuccessfully completed.

When the Batch TAIM Subsystem is activated all Statistics Records gathered since the previous Batch run are transferred to a history file and the record count in the

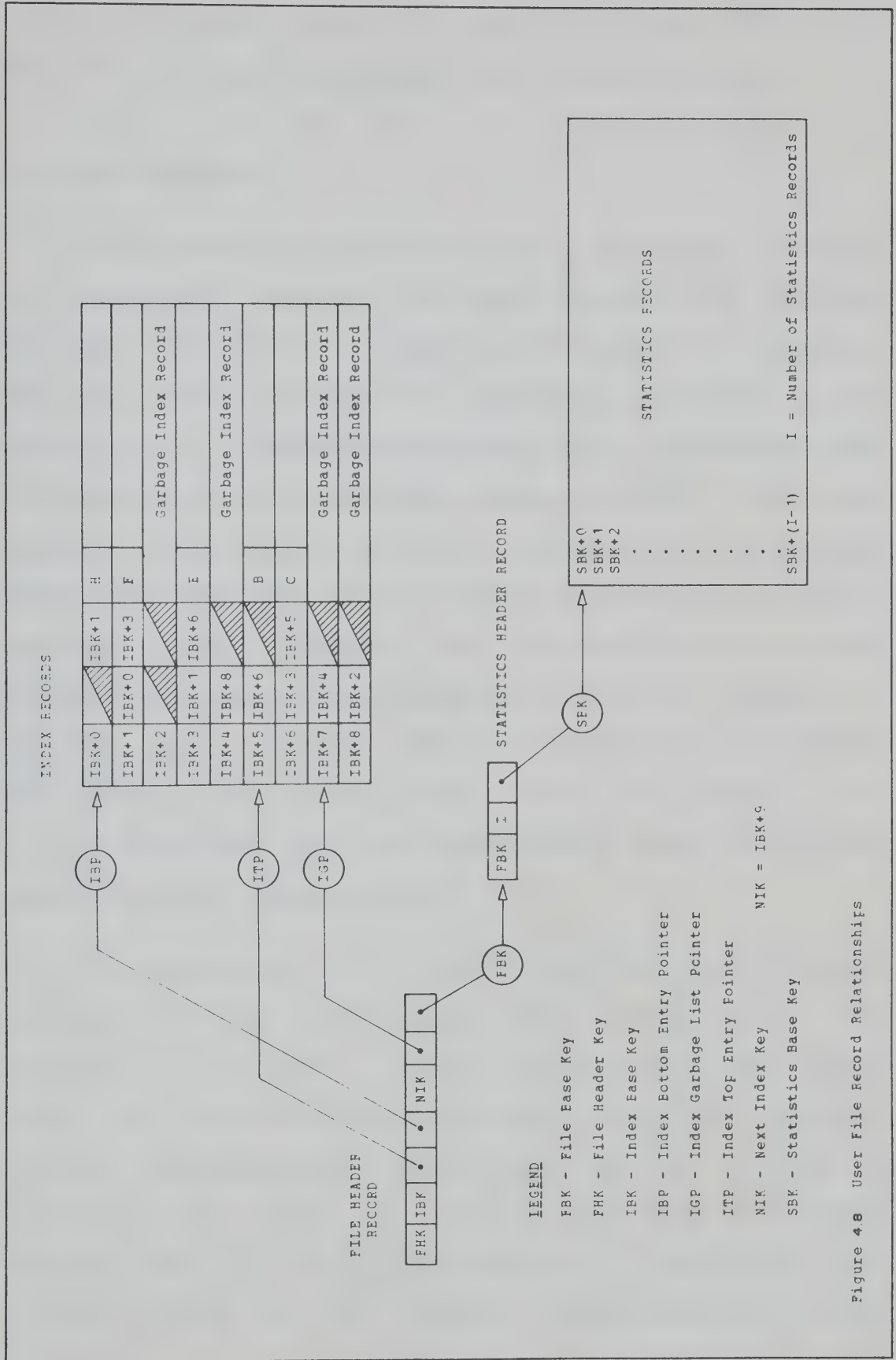


Figure 4.8 User File Record Relationships

Statistics Header Record is reset to zero. Also, the batch flag for each registered user is turned "off".

4.3 File Directory

In the TAIM files the Directory is frequently accessed and frequently altered. The Binary Search is an efficient technique for searching a Directory; however, it requires that the Index Records be physically arranged in some sequential (i.e. alphabetical) order. This requirement makes altering the Index a very time consuming affair. Insertions require moving records in order to obtain room for incoming records and deletions require moving records in order to eliminate gaps created. As a result of the above disadvantage, Directories within the TAIM files consist of "chained" Index Records. With this technique Index Records need not be in any physical order, since the logical order of the Directory is maintained by the "prior" and "post" pointers of each Index Record.

The disadvantage to chaining is that searching involves entering the Index at the logical first or last record and following the "forward" or "back" pointers until the desired record is located or determined not to be there. However, within an implementation, search times can be reduced by shortening the length of the chain that must be serially searched. That is, rather than beginning to search with the logically first or last record, begin searching at some

other pre-determined record within the chain. Since the means for determining such alternate entry points is implementation dependent, further discussion of this matter is deferred until Chapter 5 where a sample methodology is presented.

Directory control information, retained in the File Header Record, consists of: a Base Key (IBK) from which which all Index Record keys are generated; a Top Entry Pointer (ITP) which identifies the first logical record in the Directory; a Bottom Entry Pointer (IBP) which identifies the last logical record in the Directory; a Next Index Key (NIK) which contains the value of the next available Index key; and a Garbage List Pointer (IGP) which points to a list of currently unused Index Records.

Insertions

If the Directory is empty, the Index Base Key and the Next Index Key are equal and the Garbage List Pointer is not pointing. An Index Record is allocated at the Base Key location, the Top and Bottom Entry Pointers are initialized to this location, the Next Index Key is incremented and the Garbage List Pointer remains unaltered. On the other, hand, if the Directory is not empty there may or may not be records in the Directory which are currently not in use. If there exist any unused record(s), the first such record from the garbage list is removed and the Garbage List Pointer is updated to the next unused record in the list. If there are

no garbage records, a record is allocated at the next available key location and the Next Index Key is incremented. An Index Record, whether newly allocated or obtained from the garbage list, is inserted into the chain by simply adjusting "prior" and "post" pointers. If the record becomes the first logical record in the Directory, the Top Entry Pointer is updated; or, if the record becomes the last logical element in the Directory the Bottom Entry Pointer is updated; otherwise, both pointers remain unaltered.

Deletions

When an Index entry is no longer required, the record is disassociated from the chain by adjusting pointers and is placed at the top of the garbage list. The Garbage List Pointer is updated accordingly. If this record was the logical first or last record in the Directory then the respective Top or Bottom Entry Pointer is updated; otherwise, both pointers remain unaltered.

4.4 Record Key Generation

The Display, Test and Logic Files

In addition to Index control information the File Header Record contains a File Base Key (FBK) from which keys for records in the file, other than Index Records are generated.

If a new Test is to be placed in the Test File an Index Record is obtained, either allocated or taken off the garbage list, and a Unit Base Key (UBK) is generated by the equation

$$UBK = FBK + N * 1000$$

where N, the Unit Number, is calculated by subtracting the Index Base Key (IBK) from the key of the obtained Index Record. In turn, the Text Base Key (TBK), the Reference Base Key (RBK) and the Test-Key Base Key (KBK) are generated as follows :

$$TBK = UBK + 300,$$

$$TBK = UBK + 500,$$

$$KBK = UBK + 700.$$

Keys are similarly generated for Displays entering the Display File and Days entering the Logic File.

When a Display, Test or Day is no longer required, its Index Record is placed at the top of the garbage list, and by so doing, makes all keys associated with that unit available for re-use.

The Student File

The File Header Record, in addition to Directory control information, contains a Next File Key (NFK) which retains the value of the Student File's next available key. When a student is registered, a Student Leader Record is

allocated at the next available location and the Next File Key is incremented. Each time that a student encounters another Day, a new Student Packet Record is created at the next available location and is chained to the previous Day encountered, and the Next File Key is incremented. Once a key has been assigned to a student's Leader Record or to his Packet Records, there is no way of mathematically regenerating the key. Keys which have been previously assigned but which cannot be mathematically regenerated are called "Previous File Keys".

In any Student Packet, the Message Base Key (MBK), the Note Base Key (NBK), the Test Base Key (TBK) and the Display Base Key (DBK) can be generated from the key of the Student Packet Record (PFK) as follows :

$$\text{MBK} = \text{PFK} + 100,$$

$$\text{NBK} = \text{PFK} + 300,$$

$$\text{TBK} = \text{PFK} + 500,$$

$$\text{DBK} = \text{PFK} + 700.$$

CHAPTER V

TAIM - IMPLEMENTATION

5.1 Introduction

The TAIM System was implemented at the University of Alberta on an IBM 360/67 computer running under the supervision of the Michigan Terminal System (MTS). A modular approach to implementation was taken for the following reasons: (1) a modular approach prevents the overall problem from becoming too complex to implement; (2) implementation in stages is allowed for by separating functions in the system; (3) dependencies on the operating system under which the system is to operate may be confined, for the most part, to one module; (4) reasonably easy modifications are facilitated; and (5) possible expansion of the system can be accomplished with a minimum number of problems.

5.2 Interactive TAIM Subsystem

All object modules required for this Subsystem are contained in an MTS sequential file named TAIM. A brief description of each module in this file is located in APPENDIX F.

An interactive session begins when a user signs onto the MTS system according to established procedures and types in the following MTS control statement

```
$RUN TAIM
```

Upon loading the object modules MTS relinquishes control to MONITOR which begins by requesting the user to enter a six character identification code. When TAIM files, other than the System File, are not yet initialized the only identification accepted is that of the TAIM System programmer(zzzzzz); otherwise, the identification code of any registered user is honored. In the event that after three attempts the user does not enter an acceptable identification, MONITOR terminates and returns control to MTS.

Command Table

After accepting a user's identification, MONITOR establishes a relationship between a Logical I/O Unit called SYSTEM and the MTS file of the same name. Via this Logical

Unit MONITOR accesses the System File and creates a core resident command table from the Command Section Records contained therein.

Advantages of creating a command table from such an external source are: (1) command names can be easily changed; (2) abbreviation may be altered; (3) commands may be moved from one mode to another; and (4) while modifications are being made to a command in need of repair or enhancement, the command in question can be deactivated (disabled) without affecting the operation of other active (enabled) commands.

The command table produced actually contains three lists; a list of Mode 1 commands, a list of Mode 2 commands and a list of Mode 3 commands.

Logical I/O Unit Assignments

Up to this point, only one Logical I/O Unit has been assigned to a physical MTS file. Therefore, the MONITOR accesses the Online Section of the System File and from the records contained therein establishes relationships (assignments or attachments) between all other Logical I/O Units and their corresponding MTS file. Table 5.1 gives a summary of the Logical I/O Units required to operate the Interactive TAIM Subsystem.

Table 5.1 Interactive TAIM Subsystem Logical I/O Units

Logical Unit Name	Mts File Name	Description
ATTNOUT	*MSOURCE*	output to user from attention exit routine
LFILE D\$FILE	DISPLAY	Display File
HFILE H\$FILE	HOLD	Hold File
NFILE N\$FILE	NOTE	Note File
SCRATCH	-SCRATCH	scratch file
SFILE S\$FILE	STUDENT	Student File
SYSTEM	SYSTEM	System File
TERMOUT	*SINK*	output to user from all modules except attention exit routine
TERMIN	*SOURCE*	input from user
TFILE T\$FILE	TEST	Test File
TWS1	-W\$\$S\$1	scratch file for workspace 1
TWS2	-W\$\$S\$2	scratch file for workspace 2
TWS3	-W\$\$S\$3	scratch file for workspace 3
UFILE U\$FILE	USER	User File
WS1	WS1	workspace one
WS2	WS2	workspace two
WS3	WS3	workspace three

NOTE - in programming with PL/1 under MTS it is sometimes
necessary to maintain two Logical I/O Unit names
associated with the same physical MTS file

Command Recognition

In general TAIM commands consist of a command name followed by from one to five parameters, each separated by one or more blanks. When a user-defined command is entered, EXTRACT scans the input buffer in order to delineate the non-blank components of the buffer, counts the number of such components and initializes a component table. For example, if the following TAIM command were entered

POINT~~TTTT~~CLASS-~~TTTTTTTT~~BL-LOGIC.DAY

then the component table shown in Figure 5.1 would be generated.

Component	Usage	Position of First Character	Component Length
1	command	1	5
2	parameter 1	9	7
3	parameter 2	23	11
4	parameter 3	-	-
5	parameter 4	-	-
6	parameter 5	-	-

Figure 5.1 Command Buffer Component Table

The MONITOR sequentially compares the first component (command) of the input buffer to each entry in either the

Mode 1, Mode 2 or Mode 3 command lists depending on which mode is currently active. If no match is found, MONITOR informs the user that there is no such command and that it is ready to accept another command entry. However, if a match occurs, MONITOR transfers the component table and control to the appropriate work module. The routine receiving control analyzes the parameters for correctness, begins the work assignment requested and returns to MONITOR with either a zero or nonzero return code. MONITOR informs the user that the command was "DONE" or "NOT DONE" and that it it is ready to accept another command entry.

Work Module Selection

Establishing a relationship between a TAIM command and its corresponding object module can be accomplished by making the command's name identical to the module's entry point name. However, this is not a good technique since a change in the name (spelling) of a command could force the re-compilation of its corresponding module. A better approach to the problem is to permanently assign each TAIM command a unique number. Each such number, when used as an index into a list of entry point names, establishes the relationship between a command and its work module. As a result, command spellings may be changed without having to make costly re-compilations.

The EDIT Command

Within the MTS system there exists a utility text editing program called *EDIT which in addition to performing all functions of the EDIT command defined in APPENDIX A performs a variety of other extremely useful functions. Consequently the EDIT module is implemented not as a text editor but rather as an interface to the *EDIT text editing program of MTS. Subcommands recognized by the *EDIT module are summarized as follows:

- CHANGE - replace a specified string of characters with another string of characters
- DELETE - deletes specified lines from the workspace
- EDIT - allows one to edit another workspace
- INSERT - allows the insertion of lines between other lines
- LINE - causes the specified line number to become the current line number
- OVERLAY - causes lines to be overlaid with a specified string
- PRINT - allows one to list specified lines of a workspace
- RENUMBER - allows one to renumber all the lines in the workspace
- REPLACE - allows an entire line to be replaced without having to first delete then insert
- SCAN - allows one to search for the occurrence of a specified string
- SHIFT - allows lines to be shifted left or right a specified number of characters
- STOP - stop editing and return to the MONITOR

Implementing Workspaces

Within provided workspaces TAIM users create new Days, Displays or Tests and modify the contents of already existent Days, Displays or Tests. Since alterations to a workspace are performed via the *EDIT utility program and since this MTS content editor will edit only an MTS line file, TAIM workspaces (WS1, WS2, WS3) are implemented not as core-resident regions but as MTS line files.

Associated with each workspace is a "scratch pad", also an MTS line file, which is utilized if and only if the workspace contains a Day. User defined Logic Statements within the workspace are in textual form, whereas Logic Statements stored in the Logic File are in an encoded form. Therefore, workspaces contain a textual representation of Logic Statements and their corresponding scratch pads contain the encoded representation.

Resolving and Unresolving References

When a Day is taken from the Logic File and placed into a workspace each name referenced within that Day's Logic Statements is placed in one of three "old" lists, depending on whether the name is that of a Day, Display or Test.

When the Day is later returned to the Logic File there is a possibility that the Logic Statements no longer reference the same names. As a result, each name currently

referenced is placed in one of three "new" lists again depending on whether the name is that of a Day, Display or Test.

For example, if the following Logic Statements for a Day named "DAY.ONE" are placed in a workspace

```
SHOW D-DISPLAY.A
SHOW D-DISPLAY.B
TEST T-TEST.C
WHEN (T-TEST.C LT 50%) GOTO L-DAY.TWO
GOTO L-DAY.THREE
```

and, if after editing, the Logic Statements returned to the Day named "DAY.ONE" are as follows,

```
SHOW D-DISPLAY.A
SHOW D-DISPLAY.C
TEST T-TEST.C
WHEN (T-TEST.C LT 50%) GOTO L-DAY.TWO
GOTO L-DAY.FOUR
```

then Figure 5.2 shows the resultant "old" and "new" reference lists.

	Display Names	Test Names	Day Names
"old" lists	DISPLAY.A* DISPLAY.B	TEST.C* TEST.C*	DAY.TWO* DAY.THREE
"new" lists	DISPLAY.A* DISPLAY.C	TEST.C* TEST.C*	DAY.TWO* DAY.FOUR

* deleted by module UNIQUE

Figure 5.2 Lists Used for Reference Maintenance

Module UNIQUE, by deleting those names which are common to both the "old" and "new" lists, determines which names are exclusive to the "old" lists and likewise which names are exclusive to the "new" lists.

Unique (remaining) names in the "new" lists are names which were not formerly referenced but are currently referenced. The RESOLVE module then causes the name of the current Day (DAY.ONE) to be inserted into the Reference Records of each Day, Display and Test identified in the "new" lists.

Unique names in the "old" lists are names which were formerly referenced but not currently referenced. As a result the DESOLVE module is used to remove the name of the current Day (DAY.ONE) from the Reference Records of each Day, Display and Test identified in the "old" lists.

5.3 Batch TAIM Subsystem

A batch run begins when a computer operator signs onto the MTS system, again according to some established procedure, and types in the following MTS control statement

```
$SOURCE OFFLINE
```

This statement informs MTS that further directives (control statements) are to be obtained from the file named OFFLINE (see APPENDIX E). Control statements contained

therein define a job made up of the following job steps (refer to Figure 3.6):

(1) Temporary files required throughout remaining steps in the job are created.

(2) A history tape onto which statistical data will be accumulated is mounted.

(3) Modules in the file named BATCH are loaded and control is passed to the module named DRIVER. Upon receiving control, DRIVER establishes a relationship between the Logical I/O Unit called SYSTEM and the physical MTS file of the same name. Via this Logical Unit, DRIVER accesses the System File and from the Offline Section Records contained therein determines the relationships between all other Logical I/O Units and their corresponding MTS files. Table 5.2 summarizes the Logical I/O Units required for this job step. When DRIVER, in conjunction with other modules in file BATCH, has completed processing each registered student there exist unsorted teacher messages in the file named -MESGS, unsorted student printouts in the -PRINT file and unsorted TAIM commands in -BATCH.

(4) Teacher messages are sorted into ascending order by teacher within student within class within message type by a utility sort program.

(5) The object module contained in the file named PRINT

Table 5.2 Batch TAIM Subsystem Logical I/O Units

Logical Unit Name	Mts File Name	Description
BATCH	-BATCH	unsorted TAIM commands
DFILE	DISPLAY	Display File
HFILE	HOLD	Hold File
LFILE	LOGIC	Logic File
MESGS	-MESGS	unsorted teacher messages
NFILE	NOTE	Note File
OUTFILE	-PRINT	unsorted student lessons
SFILE	STUDENT	Student File
STATS	*T*	history tape
TFILE	TEST	Test File
UFILE	USER	User File

- NOTES - permanent file names consist of from one to twelve characters the first of which is not a minus sign
- temporary file names consist of from one to nine characters the first of which is a minus (-) sign
 - psuedo-device names, synonyms for true files or devices, consist of an asterisk followed by two to fourteen characters the last of which is an asterisk

is loaded and initiated. The activated MAIN module summarizes and reports the sorted teacher messages.

(6) A utility sort program arranges the student lessons into ascending order by student within class and directs the sorted lessons to a high speed printer. While sorting, the utility program calls upon the exit routine provided in file EXITA to remove the first seven characters (the sort field) from each sorted record.

(7) The utility sort program is again referenced, but this time to arrange unsorted TAIM commands into ascending order by user. For each record sorted, the utility program via the exit routine provided in EXITB, removes the first twenty-six characters. Output from this job step is directed into the MTS file named -TEMPS.

(8) In this final step, modules contained in the file named TAIM are loaded and control is relinquished to MONITOR. For each sorted command in -TEMPS, the MONITOR recognizes the command, echoes the command onto the printer and activates the appropriate work module for command execution. Since more than one user may have placed commands in the overnight batch queue, individual user printouts are separated by a TAIM banner page.

5.4 Improving Directory Search Times

In both the Interactive TAIM Subsystem and the Batch TAIM Subsystem there is extensive searching of file Directories. As was pointed out in Chapter 4, the time spent searching a chained Directory can be optimized simply by shortening the length of the chain to be searched. One approach to the problem is to create an Index Table in which a set of pointers, indexed by the characters A to Z, are created so as to point to the first Index Record in which the identification begins with the corresponding index character. Such an Index Table is shown in Figure 5.3

The use of an Index Table allows searching of the Directory to begin at some record other than the logical first or last record. For example, to determine if the Directory in Figure 5.3 contains a record identified as "EB", we index into the "E"th location of the Index Table and examine the pointer contained therein. If the pointer is null (no Index Records have identifications beginning with "E") we conclude that the required record is not in the Directory. On the other hand, if the pointer is not null (there is at least one Index Record with an identification beginning with "E") we enter the Directory at the indicated location and examine each successive record until we discover that the required record is in fact present.

The Index Table is created by the HASHNIT module, maintained by the INSERT AND DELETE modules and is referenced by the SEARCH module.

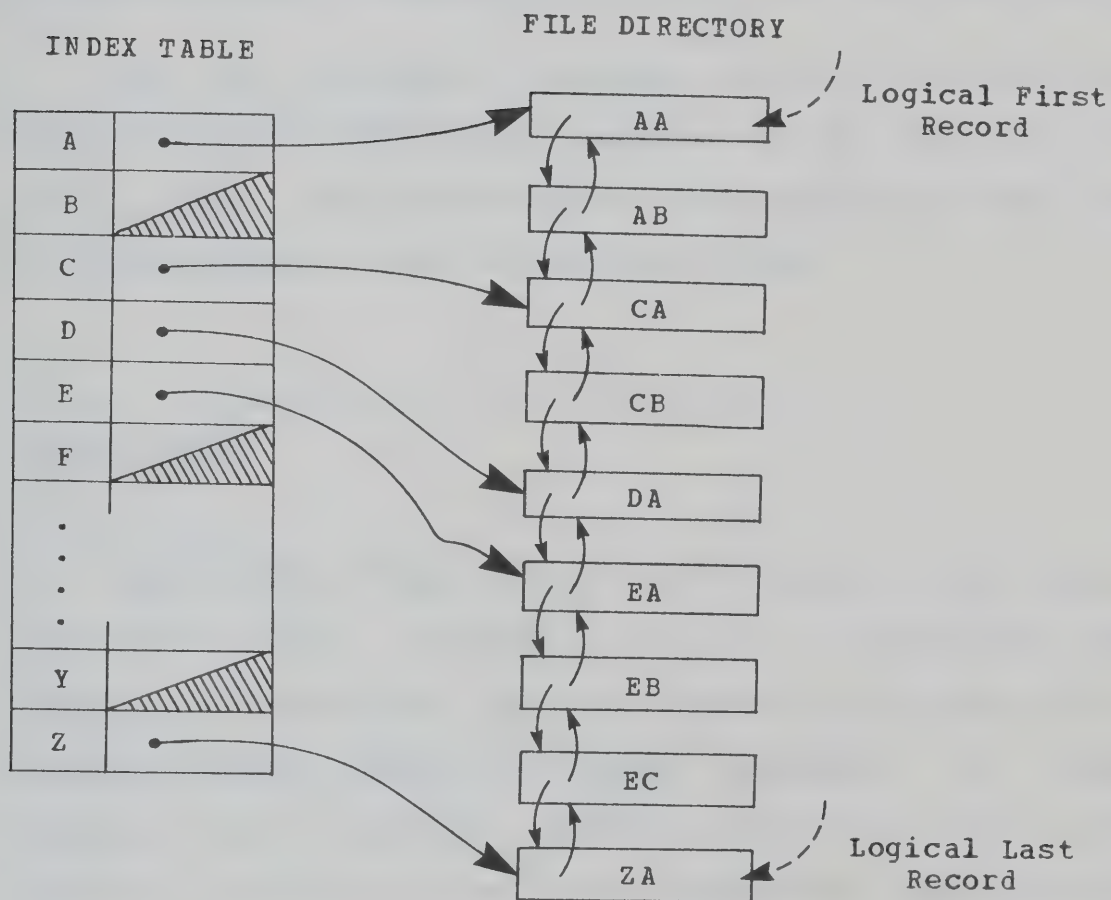


Figure 5.3 File Directory Index Table

5.5 Illustrating the TAIM System

The workings of the TAIM System will be illustrated by a series of four examples. The first three examples are terminal sessions which show most of the features of the Interactive TAIM Subsystem and the fourth example shows the lesson generating capabilities of the Batch TAIM Subsystem.

From the examples the reader should be made aware of the man-machine interaction features of the system; the power, scope and flexibility of the commands provided; and the capabilities of the entire TAIM System.

EXAMPLE ONE

The intent of this example is to show how a TAIM System programmer begins conversing with the Interactive TAIM Subsystem; initializes all files (except the System File); registers three users, a proctor (SHUNKA), an author (ZARSKY) and a teacher (WESTRM); requests a list of the currently registered users; and terminates communications with the Interactive TAIM Subsystem.

In this example, programmer-entered commands are in lower case while the computer responses are in upper case.

TAIM SYSTEM ... VERSION 1

ENTER TAIM IDENTIFICATION

ZZZZZZ

TAIM COMMENCED AT 13:21.48 ON 02-21-73

MODE 3 ACTIVATED

READY

initialize

NOT DONE ... 5

READY

why

INSUFFICIENT PARAMETERS

DONE ... \$.01

READY

init all

DONE ... \$.85

READY

why

PREVIOUS COMMAND WAS SUCCESSFULLY COMPLETED

DONE ... \$.01

READY

register shunka

NAME : dr. steve hunka

STATUS : 3

DONE ... \$.03

READY

reg shunka

DR. STEVE HUNKA IS ALREADY REGISTERED

DO YOU WISH TO MODIFY REGISTRATION? - YES OR NO

no

NOT DONE ... 30

READY

why

A "NO" RESPONSE CANCELLED THE REGISTRATION

DONE ... \$.01

READY

reg zar sky

NOT DONE ... 10

READY

why

MORE THAN 1 PARAMETER WAS OBSERVED

DONE ... \$.01

READY
 reg zarsky
 NAME : don zarsky
 STATUS : 3
 DONE ... \$.04

READY
 reg zarsky
 DON ZARSKY IS ALREADY REGISTERED
 DO YOU WISH TO MODIFY REGISTRATION? - YES OR NO
 yes
 NAME : DON ZARSKY
 NAME : okay
 STATUS : 3
 STATUS : 2
 DONE ... \$.04

READY
 reg westrom
 NOT DONE ... 15

READY
 why
 USER ID NOT 6 CHARACTERS IN LENGTH
 DONE ... \$.01

READY
 reg westrm
 NAME : marv westrom
 STATUS : 1
 DONE ... \$.03

READY
 display all
 NOT DONE ... 15

READY
 why
 FIRST PARAMETER NOT "COMMANDS" OR "USERS"
 DONE ... \$.01

READY
 disp users
 *** USER STATUS LAST-USE USES NAME
 SHUNKA 3 0 DR. STEVE HUNKA
 WESTRM 1 0 MARV WESTROM
 ZARSKY 2 0 DON ZARSKY
 DONE ... \$.03

READY
 quit

TAIM TERMINATED AT 13:30.58 ON 02-21-73

EXAMPLE TWO

This example shows how an author (ZARSKY) begins conversing with the Interactive TAIM Subsystem; builds a simplified curriculum network (Figure 5.4); registers three students (EXCEL#, FAIL##, PASS##); displays the student registrations; determines which Day (TAIM.START) the students are to encounter next; and terminates a session with the Interactive TAIM Subsystem.

In this example, author-entered commands are in lower case while computer responses are in upper case.

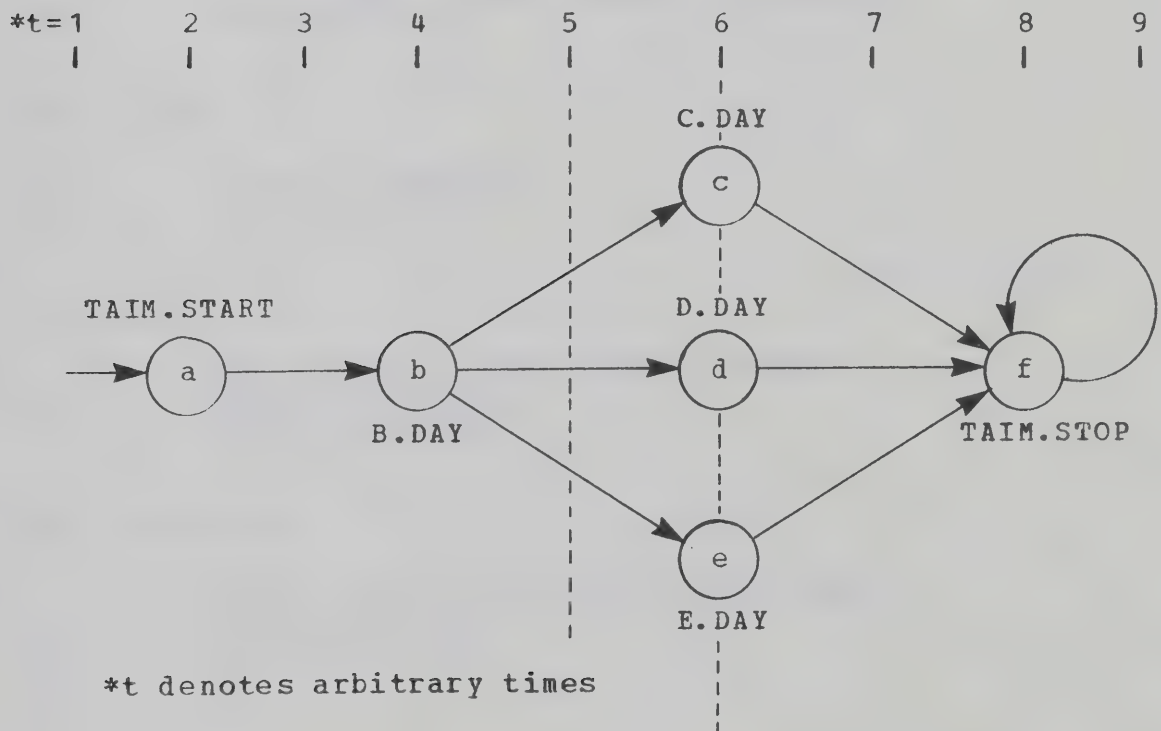


Figure 5.4 Sample Curriculum Network

The Logic Statements comprising each of the Days in Figure 5.4 are as follows:

(a) TAIM.START

```
SHOW A.PROLOGUE
GOTO B.DAY
```

(b) B.DAY

```
SHOW B.DISPLAY
TEST B.TEST
WHEN (B.TEST GE 50%) GOTO C.DAY
WHEN (B.TEST LT 50%) GOTO E.DAY
GOTO D.DAY
```

(c) C.DAY

```
SHOW CDE.DISPLAY
IF (B.TEST GE 90%) SHOW E.ENRICHMENT
GOTO TAIM.STOP
```

(d) D.DAY

```
SHOW CDE.DISPLAY
GOTO TAIM.STOP
```

(e) E.DAY

```
IF (B.TEST LT 25%) SHOW E.REVIEW
SHOW CDE.DISPLAY
GOTO TAIM.STOP
```

(f) TAIM.STOP

```
MESG "CURRICULUM NETWORK COMPLETED"
SHOW F.EPILOGUE
GOTO TAIM.STOP
```


TAIM SYSTEM ... VERSION 1

ENTER TAIM IDENTIFICATION

zarsk

UNRECOGNIZED IDENTIFICATION. TRY AGAIN

zarsky

TAIM COMMENCED AT 14:32.49 ON 02-21-73

MODE 2 ACTIVATED

READY

edit

NOT DONE ... 5

READY

why

INSUFFICIENT PARAMETERS

DONE ... \$.01

READY

edit ws 1

NOT DONE ... 10

READY

why

MORE THAN 1 PARAMETER WAS ENCOUNTERED

DONE ... \$.01

READY

edit ws1

:insert 1

?0 For the next few school days, you will be receiving

? a computer printout such as this one

?

:stop

DONE ... \$.14

READY

save ws1

NOT DONE ... 30

READY

why

WORKSPACE NOT LABELLED

DONE ... \$.01

READY

label ws1 a.prologue

NOT DONE ... 40


```

READY
why
LABEL PREFIX NOT "L-", "D-" OR "T-"
DONE ...    $.01

```

```

READY
label ws1 d-a.prologue
DONE ...    $.01

```

```

READY
save ws1
DONE ...    $.20

```

```

READY
catalog d-file
***DISPLAY LABELS
    A.PROLOGUE
DONE ...    $.02

```

```

READY
place null into ws3
DONE ...    $.08

```

```

READY
edit ws3
:insert 1
?0      The purpose of this additional brief lesson is to
? show you how logarithms can be used for solving problems
? such as compound interest value, and .....
?
:print *1
:      3      such as compound interest value, and .....
:change 3 'value'volume'
:      3      such as compound interest volume, and .....
:stop
DONE ...    $.18

```

```

READY
label ws3 ?
*** WORKSPACE NOT LABELLED
DONE ...    $.01

```

```

READY
label ws3 d-a.prologue
*** WARNING - LABEL CURRENTLY IN USE
DONE ...    $.02

```

```

READY
label ws3 d-c.enrichment
DONE ...    $.02

```


READY
 save ws3
 DONE ... \$.22

READY
 catalog d-file
 ***DISPLAY LABELS
 A.PROLOGUE
 B.DISPLAY
 C.ENRICHMENT
 CDE.DISPLAY
 E.EPILOGUE
 E.REVIEW
 F.EPILOGUE
 DONE ... \$.03

READY
 drop e.epilogue
 NOT DONE ... 25

READY
 why
 LABEL PREFIX NOT "L-","D-"OR "T-"
 DONE ... \$.02

READY
 drop d-e.epilogue
 DONE ... \$.13

READY
 catalog d-file
 ***DISPLAY LABELS
 A.PROLOGUE
 B.DISPLAY
 C.ENRICHMENT
 CDE.DISPLAY
 E.REVIEW
 F.EPILOGUE
 DONE ... \$.03

READY
 cat l-file
 ***LOGIC LABELS
 TAIM.START
 TAIM.STOP
 DONE ... \$.02

READY
 drop l-taim.start
 NOT DONE ... 30


```

READY
why
"L-TAIM.START" AND "L-TAIM.STOP" CANNOT BE DROPPED
DONE ...    $.02

```

```

READY
place l-taim.stop into ws1
DONE ...    $.19

```

```

READY
edit ws1
:print /file
:      1      GOTO L-TAIM.START
:delete *f *l
:insert 1
?nesg "curriculum network completed"
?show f.epilogue
?gototaim.stop
?
:print /file
:      1      nesg "curriculum network completed"
:      2      show f.epilogue
:      3      gototaim.stop
:stop
DONE ...    $.16

```

```

READY
eval ws1
NESG "CURRICULUM NETWORK COMPLETED"
$
*** INVALID COMMAND
GOTOTAIM.STOP
$
*** INSUFFICIENT INFORMATION
DONE ...    $.16

```

```

READY
edit ws1
:change 1 'N'M'
:      1      MESG "CURRICULUM NETWORK COMPLETED"
:change 3 'GOTO'GOTO '
:      3      GOTO TAIM.STOP
:stop
DONE ...    $.09

```

```

READY
save ws1
DONE ...    $.19

```

```

READY
place null ws2
DONE ...    $.06

```



```

READY
edit ws2
:insert 1
?if (b.test lt 25%) show e.review
?show cde.display
?goto taim.stop
?
:print /file
: 1 if (b.test lt 25%) show e.review
: 2 show cde.display
: 3 goto taim.stop
:stop
DONE ... $.12

```

```

READY
label ws2 1-e.day
DONE ... $.01

```

```

READY
save ws2
IF (B.TEST LT 25%) SHOW E.REVIEW
$
*** UNDEFINED LABEL (S)
NOT DONE ... 35

```

```

READY
why
ERRORS DETECTED DURING WORKSPACE EVALUATION
DONE ... $.02

```

```

READY
place null ws3
DONE ... $.10

```

```

READY
edit ws3
:insert 1
?0 For each of the following ten problems, you are
? offered four alternate solutions. Select whichever
? response you think is correct and place it on your
? response card .....
?
:stop
DONE ... $.26

```

```

READY
label ws3 t-b.test
DONE ... $.03

```

```

READY
key ws3
INVALID MODE 2 COMMAND

```


READY
setkey ws3
+set

RESPONSES
Q# 123456789
? 1 1
? 2 1
? 3 2
? 4 1
? 5 1
? 6 1
?

+print

										NUMBER OF QUESTIONS	6
										BETWEEN-TEST WEIGHT	1
QUESTION	RESPONSE									WEIGHTS	QUESTION
NUMBER	1 2 3 4 5 6 7 8 9									WEIGHT	
1	0 0 1 0 0 0 0 0 0									1	
2	0 1 0 0 0 0 0 0 0									1	
3	0 0 0 2 0 0 0 0 0									2	
4	1 0 0 0 0 0 0 0 0									1	
5	1 0 0 0 0 0 0 0 0									1	
6	0 0 0 1 0 0 0 0 0									1	
										WITHIN-TEST TOTAL	7

+set

RESPONSES
Q# 123456789
? 7 1
? 8 0001
? x 0010
*** INVALID QUESTION NUMBER
? 9 000x
*** INVALID WEIGHT - 4
? 9 1000
? 10 10
?

+print

										NUMBER OF QUESTIONS	10
										BETWEEN-TEST WEIGHT	1
QUESTION	RESPONSE									WEIGHT	QUESTION
NUMBER	1 2 3 4 5 6 7 8 9									WEIGHT	
1	0 0 1 0 0 0 0 0 0									1	
2	0 1 0 0 0 0 0 0 0									1	
3	0 0 0 2 0 0 0 0 0									2	
4	1 0 0 0 0 0 0 0 0									1	
5	1 0 0 0 0 0 0 0 0									1	
6	0 0 0 1 0 0 0 0 0									1	
7	0 1 0 0 0 0 0 0 0									1	
8	0 0 0 1 0 0 0 0 0									1	
9	1 0 0 0 0 0 0 0 0									1	
10	0 0 1 0 0 0 0 0 0									1	
										WITHIN-TEST TOTAL	11

+stop

DONE ... \$.44

READY
 label ws3 ?
 T-B.TEST
 DONE ... \$.01

READY
 save ws3
 DONE ... \$.23

READY
 label ws2 ?
 L-E.DAY
 DONE ... \$.01

READY
 save ws2
 DONE ... \$.34

READY
 label ws1 ?
 L-TAIM.START
 DONE ... \$.01

READY
 save ws1
 DONE ... \$.35

READY
 catalog d-file xref
 ***DISPLAY LABELS
 A.PROLOGUE
 B.DISPLAY
 C.ENRICHMENT
 CDE.DISPLAY
 E.REVIEW
 F.EPILOGUE
 DONE ... \$.04

LOGIC USING THIS LABEL
 TAIM.START
 B.DAY
 C.DAY
 C.DAY D.DAY
 E.DAY
 E.DAY
 TAIM.STOP

READY
 cat l-file xref
 ***LOGIC LABELS
 B.DAY
 C.DAY
 D.DAY
 E.DAY
 TAIM.START
 TAIM.STOP
 DONE ... \$.05

LOGIC USING THIS LABEL
 TAIM.START
 B.DAY
 B.DAY
 B.DAY
 C.DAY D.DAY
 E.DAY TAIM.STOP

READY
 catalog t-file xref
 ***TEST LABELS
 B.TEST
 DONE ... \$.01

LOGIC USING THIS LABEL
 B.DAY

READY
 drop d-cde.display
 NOT DONE ... 35

READY
 why
 THE SPECIFIED LABEL NAME IS CURRENTLY REFERENCED BY LOGIC
 DONE ... \$.02

READY
 find d-cde.display
 L-C.DAY
 L-D.DAY
 L-E.DAY
 DONE ... \$.14

READY
 find t-b.test
 L-B.DAY
 DONE ... \$.03

READY
 mode ?
 YOU ARE IN MODE 2
 DONE ... \$.02

READY
 mode 3
 NOT DONE ... 25

READY
 why
 THIS USER NOT AUTHORIZED TO ENTER REQUESTED MODE
 DONE ... \$.01

READY
 mode 1
 DONE ... \$.01

READY
 reg excel#
 NAME : john student
 AGE : 11
 SEX : male
 CLASS : a
 COMMENT : excellent student
 DONE ... \$.08


```

READY
register pass##
NAME      : helen student
AGE       : xx
*** AGE MUST BE NUMERIC
AGE       : 12
SEX       : f
CLASS     : 2
*** CLASS MUST BE A LETTER
CLASS     : b
COMMENT   : average student
DONE ...  $.09

```

```

READY
reg fail##
NAME      : sam student
AGE       : 12
SEX       : c
*** SEX MUST BE MALE OR FEMALE
SEX       : m
CLASS     : a
COMMENT   : below average student
DONE ...  $.08

```

```

READY
display all registry
EXCEL#   A   M   11   JOHN STUDENT
FAIL##   A   M   12   SAM STUDENT
PASS##   B   F   12   HELEN STUDENT
DONE ...  $.03

```

```

READY
disp excel# reg
EXCEL#   A   M   11   JOHN STUDENT
DONE ...  $.03

```

```

READY
disp pass##,excel# reg
PASS##   B   F   12   HELEN STUDENT
EXCEL#   A   M   11   JOHN STUDENT
DONE ...  $.02

```

```

READY
display class-a registry
EXCEL#   A   M   11   JOHN STUDENT
FAIL##   A   M   12   SAM STUDENT
DONE ...  $.02

```

```

READY
disp stu001 reg
NOT DONE ... 365

```


READY
 why
 STUDENT NOT REGISTERED
 DONE ... \$.01

READY
 disp excel#,stu0001 reg
 NOT DONE ... 315

READY
 why
 2ND STUDENT ID NOT 6 CHARACTERS
 DONE ... \$.01

READY
 trace class-2
 NOT DONE ... 205

READY
 why
 INVALID CLASS CODE
 DONE ... \$.01

READY
 trace class-b
 STUDENT : PASS##
 THIS LOGIC : *** REGISTERED *** DATE : 73-02-21
 NEXT LOGIC : TAIM.START
 DONE ... \$.02

READY
 trace all
 STUDENT : EXCEL#
 THIS LOGIC : *** REGISTERED *** DATE : 73-02-21
 NEXT LOGIC : TAIM.START

STUDENT : FAIL##
 THIS LOGIC : *** REGISTERED *** DATE : 73-02-21
 NEXT LOGIC : TAIM.START

STUDENT : PASS##
 THIS LOGIC : *** REGISTERED *** DATE : 73-02-21
 NEXT LOGIC : TAIM.START
 DONE ... \$.04

READY
 quit

TAIM TERMINATED AT 15:23.43 ON 02-21-73

EXAMPLE THREE

The intent of this example is to show how a teacher (WESTRM) begins a session with the Interactive TAIM Subsystem at time $t=5$ on Figure 5.4; finds out what Day (B.DAY) each student is currently at and what Test (B.TEST) each has received; reads in student responses for the Test taken; displays the marks obtained; determines which Day each student is to encounter next; delivers a notice to a student (PASS##); and terminates communications with the Interactive TAIM Subsystem.

Teacher-entered commands are in lower case while computer responses are in upper case.

TAIM SYSTEM ... VERSION 1

ENTER TAIM IDENTIFICATION

westrm

TAIM COMMENCED AT 08:30.14 ON 02-23-73

MODE 1 ACTIVATED

READY

trace all full

STUDENT : EXCEL#

THIS LOGIC : B.DAY

DATE : 73-02-22

NEXT LOGIC : *** NOT POINTED ***

*** MESSAGES

*** NOTES

*** TESTS

B.TEST

*** NOT MARKED ***

*** DISPLAYS

B.DISPLAY

STUDENT : FAIL##

THIS LOGIC : B.DAY

DATE : 73-02-22

NEXT LOGIC : *** NOT POINTED ***

*** MESSAGES

*** NOTES

*** TESTS

B.TEST

*** NOT MARKED ***

*** DISPLAYS

B.DISPLAY

STUDENT : PASS##

THIS LOGIC : B.DAY

DATE : 73-02-22

NEXT LOGIC : *** NOT POINTED ***

*** MESSAGES

*** NOTES

*** TESTS

B.TEST

*** NOT MARKED ***

*** DISPLAYS

B.DISPLAY

DONE ... \$.09

READY

mark all t-b.test

STUDENT : EXCEL#

73-02-22 ***** (*) 0.00% L-B.DAY

STUDENT : FAIL##

73-02-22 ***** (*) 0.00% L-B.DAY

STUDENT : PASS##

73-02-22 ***** (*) 0.00% L-B.DAY

DONE ... \$.03

READY
 point excel# l-stop.taim
 NOT DONE ... 110

READY
 why
 LOGIC NAME NOT FOUND
 DONE ... \$.01

READY
 point excel# taim.stop
 DONE ... \$.02

READY
 point pass##,fail## l-taim.start
 DONE ... \$.02

READY
 trace all
 STUDENT : EXCEL#
 THIS LOGIC : B.DAY DATE : 73-02-22
 NEXT LOGIC : TAIM.STOP

STUDENT : FAIL##
 THIS LOGIC : B.DAY DATE : 73-02-22
 NEXT LOGIC : TAIM.START

STUDENT : PASS##
 THIS LOGIC : B.DAY DATE : 73-02-22
 NEXT LOGIC : TAIM.START
 DONE ... \$.04

READY
 read
 --ID-- # 1234567890123456789012345678901234567890
 ?stu001
 *** STUDENT NOT REGISTERED ... INPUT IGNORED
 ?print
 --ID-- # 1234567890123456789012345678901234567890
 ?excel# x
 *** INVALID TEST NUMBER ... INPUT IGNORED
 ?excel# 2
 *** TEST "2" NOT GIVEN ... INPUT IGNORED
 ?print
 --ID-- # 1234567890123456789012345678901234567890
 ?excel# 1 22411x2413
 *** INVALID RESPONSE AT 6 ... INPUT IGNORED
 ?print
 --ID-- # 1234567890123456789012345678901234567890
 ?excel# 1 2241142413
 ?pass## 1 3321132443
 ?
 DONE ... \$.17

READY
 mark all b.test
 STUDENT EXCEL#
 73-02-22 010/011 (1) 90.90% L-B.DAY

STUDENT : FAIL##
 73-02-22 ***** (*) 0.00% L-B.DAY

STUDENT : PASS##
 73-02-22 006/011 (1) 54.50% L-B.DAY
 DONE ... \$.04

READY
 read
 --ID-- # 1234567890123456789012345678901234567890
 ?fail## 1 2331413243
 ?
 DONE ... \$.04

READY
 mark all l-b.day
 STUDENT EXCEL#
 73-02-22 010/011 (1) 90.90% T-B.TEST

STUDENT : FAIL##
 73-02-22 002/011 (1) 18.10% T-B.TEST

STUDENT : PASS##
 73-02-22 006/011 (1) 54.50% T-B.TEST
 DONE ... \$.03

READY
 trace all
 STUDENT : EXCEL#
 THIS LOGIC : B.DAY DATE : 73-02-22
 NEXT LOGIC : C.DAY

STUDENT : FAIL##
 THIS LOGIC : B.DAY DATE : 73-02-22
 NEXT LOGIC : E.DAY

STUDENT : PASS##
 THIS LOGIC : B.DAY DATE : 73-02-22
 NEXT LOGIC : D.DAY
 DONE ... \$.04

READY
 who all t-b.test
 EXCEL#
 FAIL##
 PASS##
 DONE ... \$.04

READY
who all missed t-b.test
DONE ... \$.02

READY
note excel#
ENTER NOTE : keep up the good work
DONE ... \$.03

READY
note pass##
ENTER NOTE : keep up the good work helen
DONE ... \$.03

READY
note excel#
ENTER NOTE : good work john
DONE ... \$.05

READY
disp all notes
EXCEL# -01- WESTRM "KEEP UP THE GOOD WORK"
 -02- WESTRM "GOOD WORK JOHN"
PASS## -01- WESTRM "KEEP UP THE GOOD WORK HELEN"
DONE ... \$.03

READY
unnote excel# every
DONE ... \$.02

READY
disp all notes
PASS## -01- WESTRM "KEEP UP THE GOOD WORK HELEN"
DONE ... \$.02

READY
mode 2
NOT DONE ... 25

READY
why
THIS USER NOT AUTHORIZED TO ENTER REQUESTED MODE
DONE ... \$.01

READY
quit

TAIM TERMINATED AT 09:17.38 ON 02-23-73

EXAMPLE FOUR

This example simply illustrates the lesson printouts generated for each of the registered students (EXCEL#, FAIL##, PASS##) as a result of invoking the Batch TAIM Subsystem at time $t=6$ of Figure 5.4.

EXCEL# EXCEL#
EXCEL# EXCEL#

STUDENT : JOHN STUDENT LESSON : C.DAY DATE : 73-02-23

PREVIOUS LESSON : E.DAY

TEST : B.TEST MARK : 90.90%

RESPONSES : 2 2 4 1 1 4 2 4 1 3
SCORES : 0 1 2 1 1 1 1 1 1
POSSIBLE SCORES : 1 1 2 1 1 1 1 1 1 TOTAL 10
TOTAL 11

SHOW D-CDE.DISPLAY

In the last day's lesson we learned what a log was and how to
determine its value using a graph. Today we will discuss an alternate
method

SHOW D-C.ENRICHMENT

The purpose of this additional brief lesson is to show you how
logarithms can be used for solving problems such as compound interest,
volume, and

FAIL##
FAIL##

DATE : 73-02-23

LESSON : E.DAY

STUDENT : SAM STUDENT

PREVIOUS LESSON : B.DAY

MARK : 18.10%

TEST : B.TEST

RESPONSES :	2	3	3	1	4	1	3	2	4	3
SCORES :	0	0	0	1	0	0	0	0	1	2
POSSIBLE SCORES :	1	1	2	1	1	1	1	1	1	1
										TOTAL
										11

SHOW D-E.REVIEW

You should have done better on the last day's quiz. Go back now
and review the section "LOGARITHMIC FUNCTIONS" on pages 374-376 of
your text

SHOW P-CDF.DISPLAY

In the last day's lesson we learned what a log was and how to
determine its value using a graph. Today we will discuss an alternate
method

FAIL##

FAIL##

PASS##
PASS##
PASS##
PASS##

STUDENT : HELEN STUDENT
LESSON : D.DAY
DATE : 73-02-23

PREVIOUS LESSON : B.DAY

TEST : B.TEST
MARK : 54.50%
RESPONSES : 3 3 2 1 1 3 2 4 4 3
SCORES : 1 0 0 1 1 0 1 1 0 1
POSSIBLE SCORES : 1 1 2 1 1 1 1 1 1 1
TOTAL 6
TOTAL 11

NOTE : KEEP UP THE GOOD WORK HELEN

SHOW D-CDE.DISPLAY

In the last day's lesson we learned what a log was and how to
determine its value using a graph. Today we will discuss an alternate
method

PASS##
PASS##

CHAPTER VI

RECOMMENDATIONS AND CONCLUSION

6.1 Summary

The objectives of this investigation were twofold. Firstly, to design a computer model of the Generalized TAIM System described in CHAPTER II and secondly, to implement this model on the IBM System 360/67 at the University of Alberta.

The above objectives were accomplished in the following three stages:

(1) the task of designing the file structures and defining the TAIM command language began in August 1971 and was rudimentarily completed in November of the same year.

(2) implementation and testing of the Interactive TAIM Subsystem began in November 1971 and was completed in October 1972.

(3) implementation and testing of the Batch TAIM Subsystem commenced in October 1972 and was completed near

the middle of January 1973.

At the end of the first stage, the basic model of TAIM was designed and a decision was made to use PL/1 as the implementation language. This decision was influenced by the following factors: the initial design revealed the magnitude of the system to be such that if the system were to be implemented in any reasonable length of time a higher-level language was required; and if a higher-level language is required then utilization of a language which has string handling capabilities, is self documenting, and is relatively easy to interface with other programming languages was considered desirable.

At the end of the second stage the use of the Interactive TAIM Subsystem was taught to a group of graduate and undergraduate education students who collectively constructed a one week curriculum network (Days, Displays and Tests) for a grade eleven mathematics class (Westrom [17]).

At the end of the third stage, a number of grade eleven mathematics students from a local high school were registered into the TAIM System and were administered the above developed curriculum (Westrom [17]). The Batch TAIM Subsystem was used to generate student lessons and the Interactive TAIM Subsystem was used to monitor student progress.

From the above "live" uses of the TAIM System it was determined that the design and implementation of TAIM are generally adequate. It should be noted, however, that this assumption can only be justified when the efficiency and utility of the System is ascertained after a period of controlled testing. Such testing was beyond the scope of this thesis.

7.2 Future Considerations

During the course of testing the TAIM System the following design inadequacies and implementation inefficiencies were noted:

Student File Directory

The Student File Directory contains Index Records (one for each registered student) logically ordered by student identification. The problem with having such a Directory is that determining which students belong to a particular class requires that the class letter of every Index Record be compared to the letter of the class in question. It is apparent that as the number of registered students increases, the above procedure will become more inefficient. Future versions of TAIM might overcome this inefficiency by incorporating:

- (1) an additional pointer in the File Header Record

- (2) a Student Class List Record
- (3) an additional pointer in each Student Index Record.

The new field in the File Header Record should simply point to the Student Class List Record, which should contain a key followed by twenty-six pointers (one for each of the class codes A-Z). Since the additional field on each Index Record can be used to link Directory records into class chains, each of the twenty-six fields on the Student Class List Record (see Figure 6.1) should point to the logical first record of the corresponding class. GO

With such a technique the Directory, in addition to being in logical order by student identification, is in logical order by student within class.

Dynamic Loading of Modules

All modules comprising the Interactive TAIM Subsystem (see APPENDIX F) are loaded into core at once - a total storage requirement of approximately 275K (200K for the TAIM modules plus 75K for the PL/1 library routines). In future implementations, it may be worth grouping individual modules into small sections and dynamically loading and unloading each section as required. In addition, all coding should be made re-entrant so that core-resident modules can be used by several users simultaneously. Although this will result in lower storage costs, it must be remembered that the reduction in cost due to smaller memory requirements must be

carefully weighed against increases in cost due to CPU time required to load and unload program sections in question.

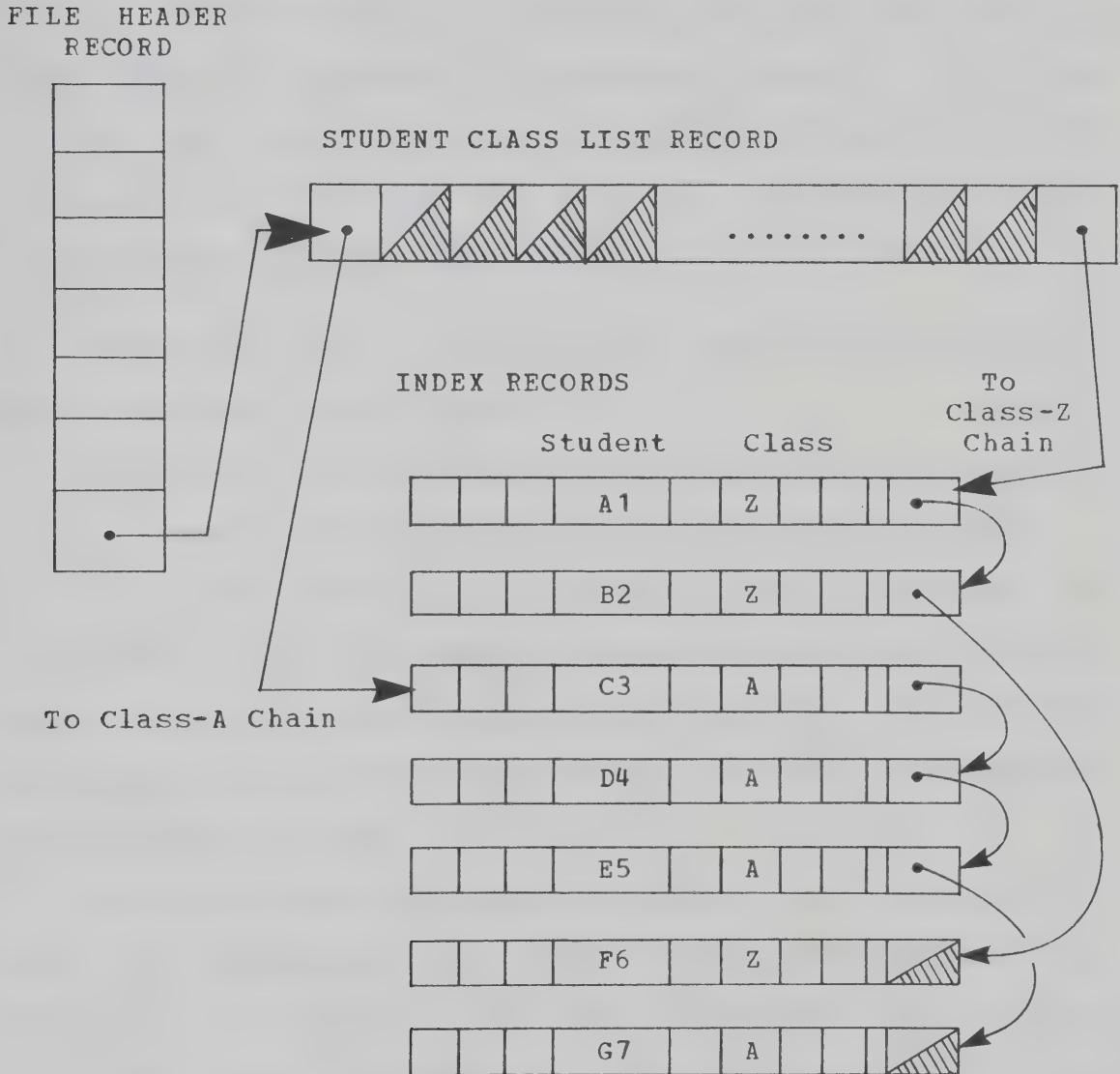


Figure 6.1 Chaining the Student Directory by Class

Saving File Index Tables

Each time that either the Interactive TAIM Subsystem or the Batch TAIM Subsystem is invoked a Directory Index Table, one for each file with a Directory, needs to be created. At present, Directories are relatively small and thus creating Index Tables is relatively inexpensive. However, this may not be the case if and when larger Directories come into being. Should the cost of creating such tables become prohibitive a possible solution might be as follows:

(1) provide space, in each file containing a Directory, for a Directory Index Table.

(2) when a file containing a Directory is initialized create a corresponding Index Table in the space provided.

(3) each time that the Interactive TAIM Subsystem or the Batch TAIM Subsystem is invoked transfer all Directory Index Tables into core storage where they can be directly maintained by the INSERT and DELETE routines and referenced by the SEARCH routine.

(4) the Interactive TAIM Subsystem both modifies as well as references the Index Tables. Therefore, upon terminating an invocation of this Subsystem each updated Index Table should be returned to the file from which it was originally obtained. The Batch TAIM Subsystem references but does not modify the Index Tables. Therefore, upon terminating an invocation of this Subsystem the Index Tables need not be saved.

Optimizing Record Lengths

As an aid in debugging, the current implementation retains numeric values, in particular record keys and pointers, in an external binary coded decimal representation. Two disadvantages in doing so are: disk storage is wasted and unnecessary conversions from external to internal form and vice versa take place during processing. To overcome these disadvantages future implementations might consider retaining numeric values in an internal binary coded form thereby optimizing disk storage and recovering time spent in unnecessary conversions.

7.3 Concluding Remark

The model developed for this thesis, while being operative as it presently exists, can be used as a foundation upon which further research, in the area of utilizing TAIM as an instructional manager, might be based.

There is one question in particular, which future research must answer - "In a large scale implementation, will the basic design of the system and its associative files be able to cope with a prolific growth of data?".

REFERENCES

- [1]. Baker, F.B., "Computer-Based Instructional Management Systems: A First Look", Review of Educational Research, February, 1971.
- [2]. Connolly, J.A., and T.F. Lampert, "Computer Managed Instruction: IBM System 360", Behavioral Research Methods and Instruction, February, 1972.
- [3]. Cooley, W.W. and R. Glaser, "An Information and Management System for Individually Prescribed Instruction", Working Paper No. 44, Learning Research and Development Center, University of Pittsburgh, 1968.
- [4]. Flanagan, J.C., "Program for Learning in Accordance with Needs", Psychology in the Schools, June, 1969.
- [5]. Flores, I., Data Structure and Management, Englewood Cliffs, N.J., Prentice-Hall, Inc., 1970.
- [6]. Gauthier, R.L. and S.D. Ponto, Designing Systems Programs, Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1970.
- [7]. Guide to Writing a Terminal Monitor Program or a Command Processor, IBM Corporation, New York, N.Y., Form GC28-6764, 1971.
- [8]. Katzan, H., Advanced Programming: Programming and Operating Systems, New York, N.Y., Van Nostrand Reinhold Co., 1970.

- [9]. Kelly, A.C., "An Experiment with TIPS: A Computer Aided Instructional System for Undergraduate Education", The American Economic Review, October, 1968.

- [10]. Knuth, D.E., The Art of Computer Programming, Vol. 1, Fundamental Algorithms, Reading, Massachusetts, Addison-Wesley Publishing Co., 1968.

- [11]. Michigan Terminal System, Vol. 3, System Subroutines, University of Alberta, Edmonton, Department of Computing Services, 1972.

- [12]. PL/1 Subroutine Library Logic Manual, IBM Corporation, New York, N.Y., Form GY28-6801, 1969.

- [13]. Salisbury, A.B., "Computers and Education: Toward Agreement on Terminology", Educational Technology, September, 1971.

- [14]. Sayers, A.P. (Ed), Operating Systems Survey, New York, N.Y., Auerbach Publishers Inc., 1971.

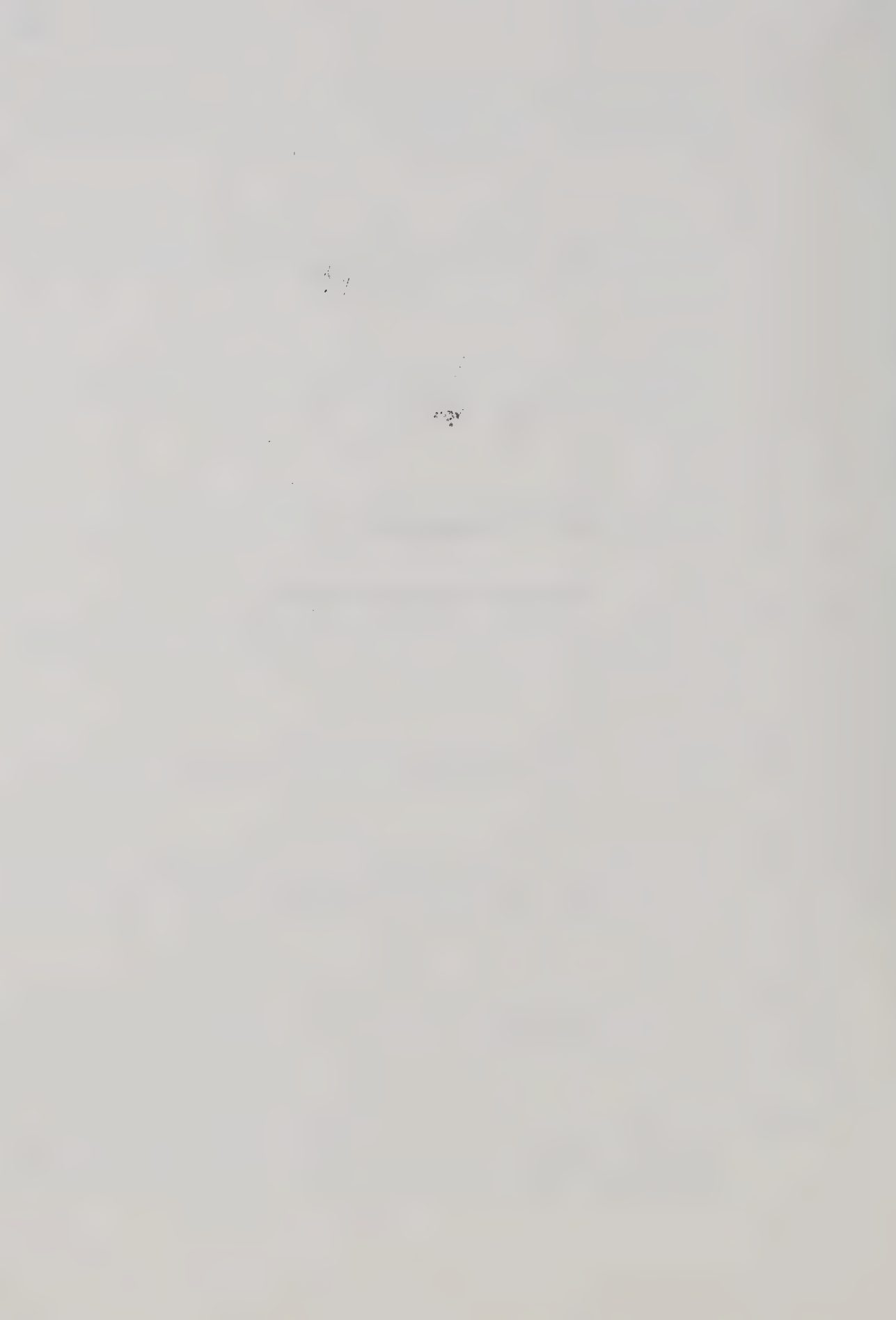
- [15]. Silberman, H.G., Design Objectives of the Instructional Management System, Systems Development Corp., Santa Monica, California, SP 3038/001/00, 1968.

- [16]. Time Sharing Option Guide, IBM Corporation, New York, N.Y., Form GC28-6698, 1971.

- [17]. Westrom, M.L., "The Teacher-Authored Instruction Manager (TAIM) : A Computer Managed Instruction System", Ph.D. Thesis, University of Alberta, Edmonton, 1973.

APPENDIX A

INTERACTIVE TAIM COMMANDS



The following notation is used to describe the TAIM command language:

UPPER CASE	must appear as shown
lower case	replace appropriately
	separates alternate forms
{ }	encloses alternate forms
[]	encloses optional forms

NOTE - The sids parameter used in TAIM commands may take any of the following three forms:

- (1) "ALL" meaning all registered students,
- (2) "CLASS-letter" meaning all students in the indicated class, or
- (3) a list of up to five student identifiers separated by commas.

NOTE - Each command is identified by A-## or I-## where A indicates that the command is currently implemented (Active), I indicates that the command is currently not implemented (Inactive) and ## is a number that has been assigned to the command.

SYNTAX AND DESCRIPTION OF MODE 1 COMMANDSAVERAGE sids [FULL]

A-41

Upon entering this command, the user is asked to identify the Tests over which a weighted average is to be calculated. The Test names entered form a check list against which is compared the names of the Tests that an individual student has taken. As a student is processed, each test that the student has taken, beginning with the most recent, is compared to the check list. If a match occurs, the name is removed from the check list and the marks obtained by the student on that test are retained for later averaging. If no match occurs the student's next Test is retrieved, and the check list is scanned again. Averaging calculations begin when either there are no more student tests to be compared to the check list (check list is not empty), or a mark has been recovered for each of the specified Tests (check list is empty). Individual displays consist of: (1) the student's identification; (2) a printed line, for each recovered Test, which contains the Test's name, the date the Test was given, the mark received, the test weight and if the "FULL" option was specified the name of the Day which gave the Test; and (3) the weighted average taken over all Tests listed in (2).

DISPLAY sids {HOLDS|NOTES|MESSAGES|REGISTRY}

A-07

For the indicated range of students the user can:

- (1) determine who is currently under suspension (see HOLD command);
- (2) display any pending notes (see NOTE command);
- (3) display any messages directed to the teacher during the previous night's BATCH run (see MESG Logic Statement); and
- (4) display available student registration data (see REGISTER command).

HOLD sids #days

A-03

The indicated students are suspended for the specified number of school days. This simply means that for the specified number of BATCH runs, lesson printouts will not be generated for these students.

MARK sids [FULL] {L|T}-name

A-04

When T-name is specified, marks obtained by the indicated students on the named Test are retrieved and displayed. If an individual student has taken the same Test more than once, the most recent results are displayed first.

When L-name is specified, then for each of the indicated students, marks of all tests taken within that Day are retrieved and displayed. If an individual student has encountered the same Day more than once, the most recent Day results are displayed first.

If "FULL" is specified, then not only is the test mark displayed, but the actual student responses from which the mark was derived are also displayed.

MODE {1|2|3|?}

A-05

Allows the user to remain in Mode 1, enter Mode 2 if she is an author or proctor, enter Mode 3 if and only if she is registered as a proctor, or to display the currently active mode.

NOTE sids [[D-]name]

A-01

If the name of a Display containing a notice is not specified, the user is prompted for a literal notice. The notice name or literal notice is placed in one of several queues and remains there until the next BATCH run. The queue into which the notice is retained depends on the sids parameter used. If "ALL" is specified, the notice is placed in the "ALL" queue; if "CLASS-A" is specified then the notice is retained in the "CLASS-A" queue (there are twenty six such queues - one for each class); or if individual student identifications are specified then the notice is retained in each individual's dynamically allocated queue. During the next BATCH run the queued notices are recovered and directed to the indicated students.

POINT sids [L-]name

A-06

The next lesson pointer for each of the indicated students is set to that of the named Day.

QUIT

A-13

Interaction with the command MONITOR is terminated.

READ

A-02

Activates an interactive routine which reads student responses, marks the responses according to an answer key and interprets appropriate Decision Logic Statements to determine which Day the student should next receive.

REGISTER student

A-08

This command activates an interactive routine which prompts the user for the indicated student's name, age, sex, and class; and for any pertinent comments relating to that student. If the student is already registered, modifications to his registration data are permitted.

TRACE sids [FULL] #days

A-10

For each student, a list of the Days encountered during the preceding number of school days is displayed. The "FULL" option causes all messages, notes, Test names and Display

names within each Day listed to also be displayed.

UNHOLD sids

A-37

Releases the named student's from any pending suspensions (see HOLD command).

UNNOTE sids {note#|EVERY}

A-36

Notices in a queue are numbered 1 thru n, where n is the number of the last notice. This command removes a notice from a queue and renumbers those remaining. The queue or queues from which the notice is to be removed is identified by the sids parameter, and the notice itself is identified by a positive number relative from one. If "EVERY" is specified, then all notices in the indicated queue are removed (i.e. queue becomes empty).

UNREGISTER student

A-09

The student's registration is deleted.

WHEN sids {D|L|T}-name

I-42

Displays the date(s) on which the indicated Display, Day or Test was encountered. If the second parameter specifies a Display or Test then the name of the Day, from which that Display or Test was administered, is also displayed.

WHO sids [MISSED] {D|L|T} -name

A-11

Selects from the indicated students those who have encountered (or did not encounter) the specified Day, Display or Test.

WHY

A-12

If the execution of any Mode 1 command is abnormally terminated a numeric error code reflecting the cause of termination is displayed. With this command, the user requests that an explanation of the error code be given.

SYNTAX AND DESCRIPTION OF MODE 2 COMMANDS

CATALOG {D|L|T}-FILE [XREF]

A-43

Used to print, in alphabetical order, the names of any current Displays in the Display File, Days in the Logic File or Tests in the Test File. If the optional "XREF" parameter is specified, then following each name displayed is a list of all Days whose Logic Statements reference that name.

DROP {D|L|T}-name

A-16

A Display, Day or Test may be permanently removed from the TAIM files provided that the named unit exists and is not referenced by any Logic Statement within any Day.

EDIT ws{1|2|3}

A-17

This command is utilized whenever the contents of one of three available workspaces is to be modified. After the command is entered and the workspace to be modified is identified, the MONITOR is prepared to receive any of the following commands:

DELETE - deletes an existent line from the workspace
 INSERT - inserts a new line into the workspace
 PRINT - displays the contents of a line

- REPLACE - replace the contents of an already existent line
- STOP - informs the MONITOR that no further editing subcommands are to be entered at this time

EVALUATE WS{1|2|3}

A-18

Before the contents of a labelled workspace (see LABEL command) can be permanently stored in either the Display File, Logic File or Test File, the contents of the workspace must satisfy certain minimum requirements. This command is then used to determine if the workspace contents meet the minimum requirements of the destination file. The minimum file requirements are as follows:

DISPLAY FILE REQUIREMENTS

- there must be at least one line of text

LOGIC FILE REQUIREMENTS

- there must be at least one Logic Statement
- all Logic Statements must be syntactically correct
- there may be only one GOTO statement and it must be the last
- a WHEN statement must not be followed by any IF, MESH, SHOW or TEST statements
- all labels used must currently exist

TEST FILE REQUIREMENTS

- there must be at least one question.
- there must be an answer key (see SETKEY command) for the defined Test

EXECUTE ws{1|2|3}

I-15

The effect of executing a labelled workspace (see LABEL command) depends entirely on the workspace content at the time of execution.

If the workspace contains a Display there is no effect.

If the workspace contains a Test, the user is asked to give a response to each question in the Test. When a question is answered, the response is graded and a mark is displayed. The user can thus determine if the Test is being properly marked.

If the workspace contains a Day, the interpretation of Logic Statements therein commences. As each statement is interpreted, the user is requested to give "VALUE" values, and the resultant action is displayed. The user can thus scrutinize the actions of a Day.

FIND {D|L|T}-name

A-19

A Day contains Logic Statements which reference Displays, Tests and Days. It is sometimes useful to know where these references occur. Therefore, this command prints the names of all Days which reference the indicated Display, Day or Test.

LABEL ws{1|2|3} {?|{D|L|T}-name}

A-21

Before the contents of a workspace can be evaluated or saved the workspace must be labelled. By using this command the user can assign a label to a workspace or can find out what a current workspace label is. The assigned label not only gives a name to the workspace contents, but also identifies the contents of the workspace as being either a Day, Display or Test.

MODE {1|2|3|?}

A-22

This command allows the user to unconditionally enter Mode 1, enter Mode 3 if and only if she is registered as a proctor, remain in Mode 2, or find out what mode is currently active.

PLACE {ws{1|2|3}|NULL|{D|L|T}-name} [INTO] ws{1|2|3}

A-23

With this command the user may: (1) empty a workspace (workspace becomes unlabelled); (2) copy the contents of one workspace into another workspace (workspace label is also transferred); or place a copy of the indicated Day, Display or Test into a workspace (workspace assumes label of the indicated Day, Display or Test).

QUIT

A-28

Interaction with the Command MONITOR is terminated.

PENAME {D|L|T}-name [AS] newname

I-14

The current name of the specified Display, Day or Test and all occurrences of this name within the TAIM files are replaced by the indicated "new" name.

SAVE ws{1|2|3}

A-26

The contents (Day, Display or Test) of the indicated workspace is stored in either the Logic File, Display File or Test File provided that certain minimum requirements are satisfied (see EVALUATE command). The file into which the workspace contents are stored and the name by which it shall be known depends on the label assigned to the workspace (see LABEL command).

SETKEY ws{1|2|3}

A-27

This command allows the user to define an answer key for the Test contained in the indicated workspace. After the command has been entered and the workspace containing the Test is identified, the monitor is prepared to receive any of the following subcommands: CLEAR, PRINT, SET, STOP, AND WEIGHT.

CLEAR - clears the answer key

PRINT - prints the current status of the answer key

- SET - initiates a prompting procedure which allows the user to define or modify response weights for any question in the key
- STOP - informs the MONITOR that further subcommands are not to be entered at this time
- WEIGHT - allows the user to specify a test weight for the answer key

WHY

A-24

If the execution of any Mode 2 command is abnormally terminated a numeric error code reflecting the cause of termination is displayed. With this command, the user requests that an explanation of the numeric code be given.

SYNTAX AND DESCRIPTION OF MODE 3 COMMANDS

CONDENSE {ALL|{D|H|L|N|S|T|U}-FILE}

I-40

In order to improve execution and response times, the command MONITOR does only a minimum amount of garbage collection within the following TAIM files: Display File, Hold File, Logic File, Note File, Student File, Test File and the User File. As a result, after a period of operation, the files accumulate a number of redundant records. With this command, a proctor may repack all of the TAIM files at once or one by one as desired.

DISPLAY {COMMANDS [1|2|3]| USERS}

A-33

With this command, the user can: (1) display the names abbreviations, status and assigned number of all TAIM commands, Mode 1 commands, Mode 2 commands or Mode 3 commands; or (2) display the identification, name, responsibility level and usage data (date registered, date of first use, date of last use, total number of uses) for each registered user.

DUMP {ALL|{D|H|L|N|S|T|U}-FILE}

I-29

The task of maintaining a TAIM file is simplified if users have access to a comprehensive, up to date, formatted

listing of a file's contents. With this command, a proctor can produce such listings for all files at once, or for specific files one at a time.

INITIALIZE {ALL|{D|H|L|N|S|T|U}-FILE}

A-35

Before the system can be totally utilized a privileged proctor must sign-on and generate (initialize) the following seven TAIM files: Display File, Hold File, Logic File, Note File, Student File, Test File and the User File. The privileged proctor can initialize all of the above files at once, or she can initialize them one by one as desired.

MODE {1|2|3|?}

A-32

Any user who is in mode 3 must be a proctor, therefore, with this command the user can unconditionally enter modes 1 or 2, remain in mode 3, or have the currently active mode displayed.

MTS [\$]mts-command

A-39

TAIM runs under a time sharing supervisor called the Michigan Terminal System (MTS). With this command, the user requests that the supervisor interrupt TAIM processing, execute the given MTS command and continue processing after the command has been executed.

The execution of several MTS commands will have catastrophic effects on the TAIM system and should never be requested. They are: RUN, START, RESTART, LOAD, UNLOAD, SIGNON, and SIGNOFF.

OFFLINE

A-38

Oftentimes, output produced at a terminal is voluminous. With this interactive command the user at a remote terminal, places TAIM commands into a specially provided queue for Batch processing. That night at the data center, commands in this queue are executed and the subsequent output is directed to a high speed printer. From the data center, the printout is dispatched for morning delivery to the user.

QUIT

A-34

Interaction with the Command Monitor is terminated.

REGISTER user

A-30

A mode 3 user (proctor) may register other users into the TAIM system. This command, activates an interactive procedure which prompts the registrar for the registrant's name and level of responsibility (status). A person's status can be that of a mode 1 user (teacher); mode 2 user (author); or a mode 3 user (proctor).

SET command# {ACTIVE|INACTIVE}

A-20

Very often the resultant actions of a particular command are erroneous or are in some need of enhancement. In either case, rather than making the entire Interactive TAIM Subsystem unavailable for the duration of any necessary changes, it is preferable to temporarily de-activate the command being serviced and allow remaining commands in the system to function normally. When modifications have been completed, the command may be activated. With this command, the user can dynamically and immediately render a TAIM command active (available) or inactive (unavailable). The command in question is identified by it's assigned command number and not by name. Rendering an already active command active has no effect as does rendering an already inactive command inactive.

UNREGISTER user

A-31

The specifies user's registration is cancelled.

WHY

A-25

If the execution of any mode 3 command is abnormally terminated a numeric error code is displayed. With this command, the user requests that an explanation of the numeric code be given.

APPENDIX B

DAY (LOGIC) CONSTRUCTION SYNTAX

The special BNF symbols used in defining a Day are given in the following table:

Symbol	Means
::=	is defined to be
	or
< >	a name

NOTE - Conditions within the current version of TAIM are limited to comparing some threshold value with : (1) final Test marks; (2) individual question responses; and (3) counts of the number of times a student has passed through a particular Day of the curriculum network.


```

<day> ::= <lesson part><decision part>

<lesson part> ::= <null>|<lesson statement>|<lesson statement><lesson part>
<lesson statement> ::= <show statement>|<test statement>|<message statement>|<if statement>
<show statement> ::= SHOW <display label>
<test statement> ::= TEST <test label>
<message statement> ::= MSG "<character sequence>"
<if statement> ::= <if specification><show statement>|<if specification><message statement>
<if specification> ::= IF <condition>

<decision part> ::= <when group><goto statement>
<when group> ::= <null>|<when statement>|<when statement><when group>
<when statement> ::= WHEN <condition><goto statement>
<goto statement> ::= GOTO <day label>

<condition> ::= (<single condition>|<double condition>)
<double condition> ::= <single condition><conjunction><single condition>
<single condition> ::= <mark condition>|<response condition>|<count condition>
<mark condition> ::= <test label><relation><mark>%|<test label><relation><mark>

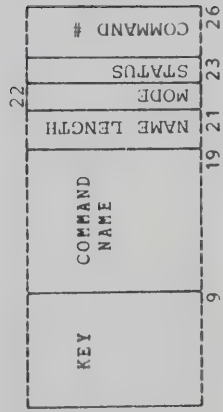
```


<response condition> ::= <question identifier><relation><digit>R
 <count condition> ::= L-<name><relation><count>
 <mark> ::= <digit sequence>
 <count> ::= <digit sequence>
 <test label> ::= T-<name>|<name>
 <display label> ::= D-<name>|<name>
 <day label> ::= L-<name>|<name>
 <question identifier> ::= <test label>(<question number>)
 <question number> ::= 1|2|3|4|5|6|7|8|9|10|11| |38|39|40
 <name> ::= from 1 to 20 <character> the first of which is <letter>
 <character> ::= <letter>|<any non-letter character>
 <character sequence> ::= from 1 to 100 <character>
 <letter> ::= A|B|C|D| |X|Y|Z
 <digit> ::= 0|1|2|3|4|5|6|7|8|9
 <relation> ::= LT|LE|EQ|NE|GE|GT
 <conjunction> ::= AND|OR
 <digit sequence> ::= from 1 to 4 digits

APPENDIX C

RECORD LAYOUTS

COMMAND SECTION RECORD



LOGIC SECTION RECORD



DISPLAY SECTION RECORD



TEST SECTION RECORD

KEY	ANY TEST FILE RECORD
-----	----------------------

9 Max 255

STUDENT SECTION RECCRD

KEY	ANY STUDENT FILE RECORD
-----	-------------------------

9 Max 255

USER SECTION RECORD

KEY	ANY USER FILE RECORD
-----	----------------------

9 Max 255

HOLD SECTION RECORD

KEY	ANY HOLD FILE RECORD
9	Max 255

NOTE SECTION RECORD

KEY	ANY NOTE FILE RECORD
9	Max 255

ERROR SECTION RECORD

KEY	BASE	3
	COMMAND#	6
	REFCODE	9
	- ERROR MESSAGE TEXT	
	Max 255	

ONLINE SECTION RECORD

KEY	INTERACTIVE (ONLINE) SUBSYSTEM'S LOGICAL I/O ASSIGNMENTS	Max 255
-----	--	---------

OFFLINE SECTION RECORD

KEY	BATCH (OFFLINE) SUBSYSTEM'S LOGICAL I/O ASSIGNMENTS	Max 255
-----	---	---------

FILE HEADER RECORD

KEY	INDEX BASE KEY	INDEX TCF ENTRY PCINTER	INDEX BOTTOM ENTRY POINTER	NEXT INDEX KEY	INDEX GARAGE LIST POINTER	FILE BASE KEY
9	18	27	36	45	54	63

INDEX RECORD

STUDENT LEADER RECORD	CLASS CODE	HOLD RECORD POINTER	NOTE RECORD POINTER	READ FLAG
57	56	66	75	76

Student File Only

KEY	NEXT INDEX POINTER	BACK INDEX PCINTER	INDEX IDENTIFICATION	UNIT LEADER RECORD POINTER
9	18	27	47	56

User File Only

USER STATUS	REGISTRY DATE	FIRST USE DATE	LAST USE DATE	NUM S E S	BATCH FLAG	NAME LENGTH	VARIABLE LENGTH NAME
48	56	64	72	76	77	79	Max 119

STUDENT LEADER RECFD

KEY	9	FIRST PACKET POINTER	18	LAST PACKET POINTER	27	REGISTRY DATE	35	AGE	37	SEX	39	CLASS	41	NAME LENGTH	44	COMMENT LENGTH	44	VARIABLE LENGTH NAME										VARIABLE LENGTH COMMENT										Max 184									
-----	---	----------------------------	----	---------------------------	----	------------------	----	-----	----	-----	----	-------	----	-------------	----	-------------------	----	----------------------	--	--	--	--	--	--	--	--	--	-------------------------	--	--	--	--	--	--	--	--	--	---------	--	--	--	--	--	--	--	--	--

STUDENT PACKET RECORD

KEY	9	NEXT PACKET POINTER	18	BACK PACKET PCINTER	27	CURRENT LOGIC IDENTIFIER	47	NEXT LOGIC IDENTIFIER	67	DATE	75	TIME	83	MESSAGE BASE KEY	92	NUM MESSAGE RECORDS	96	NOTE BASE KEY	105	NUM NOTE RECORDS	109	TEST BASE KEY	118	NUM TEST RECORDS	122	DISPLAY BASE KEY	131	NUM DISPLAY RECORDS	135	TOTAL NUM DISPLAYS	139
-----	---	---------------------------	----	---------------------------	----	-----------------------------	----	--------------------------	----	------	----	------	----	------------------------	----	------------------------	----	---------------------	-----	---------------------	-----	---------------------	-----	---------------------	-----	------------------------	-----	------------------------	-----	-----------------------	-----

STUDENT MESSAGE RECORD

KEY	MESSAGE CODE	MESSAGE LENGTH	VARIABLE LENGTH MESSAGE
9	10	13	Max 113

- Message Codes
- 1 - Not Used
 - 2 - Display Name
 - 3 - Not Used
 - 4 - Literal

STUDENT NCTE RECORD

KEY	NOTE SENDER	NOTE CODE	NOTE LENGTH	VARIABLE LENGTH NOTE
9	30	29	33	Max 133

- Note Codes
- 1 - Not Used
 - 2 - Display Name
 - 3 - Not Used
 - 4 - Literal

STUDENT TEST RECORD

KEY	TEST IDENTIFIER	MARK SWITCH	TEST WEIGHT	TEST MARK	TEST TOTAL	TEST PERCENTAGE	NUMBER OF QUESTIONS	RESPONSE	SCORE	POSSIBLE	Q #1	Q #2	Q #3	----- up to 40 Questions per record -----
9	29	31	34	37	42	45	48	48	48	48				Max 165

STUDENT DISPLAY RECORD

KEY	DISPLAY IDENTIFIER	DISPLAY IDENTIFIER	----- up to 10 Display Names per record -----
9	29	29	Max 209

UNIT LEADER RECCRD

KEY	TEXT/LOGIC BASE KEY	NUM TEXT	REFERENCES BASE KEY	NUM REF RECORDS	TOTAL NUM REFERENCES	TEST-KEY BASE KEY	NUM TEST-KEY RECORDS
9	18	22	31	35	39	48	52

TEXT RECORD

KEY	VARIABLE LENGTH TEXT
9	Max 255

REFERENCE RECORD

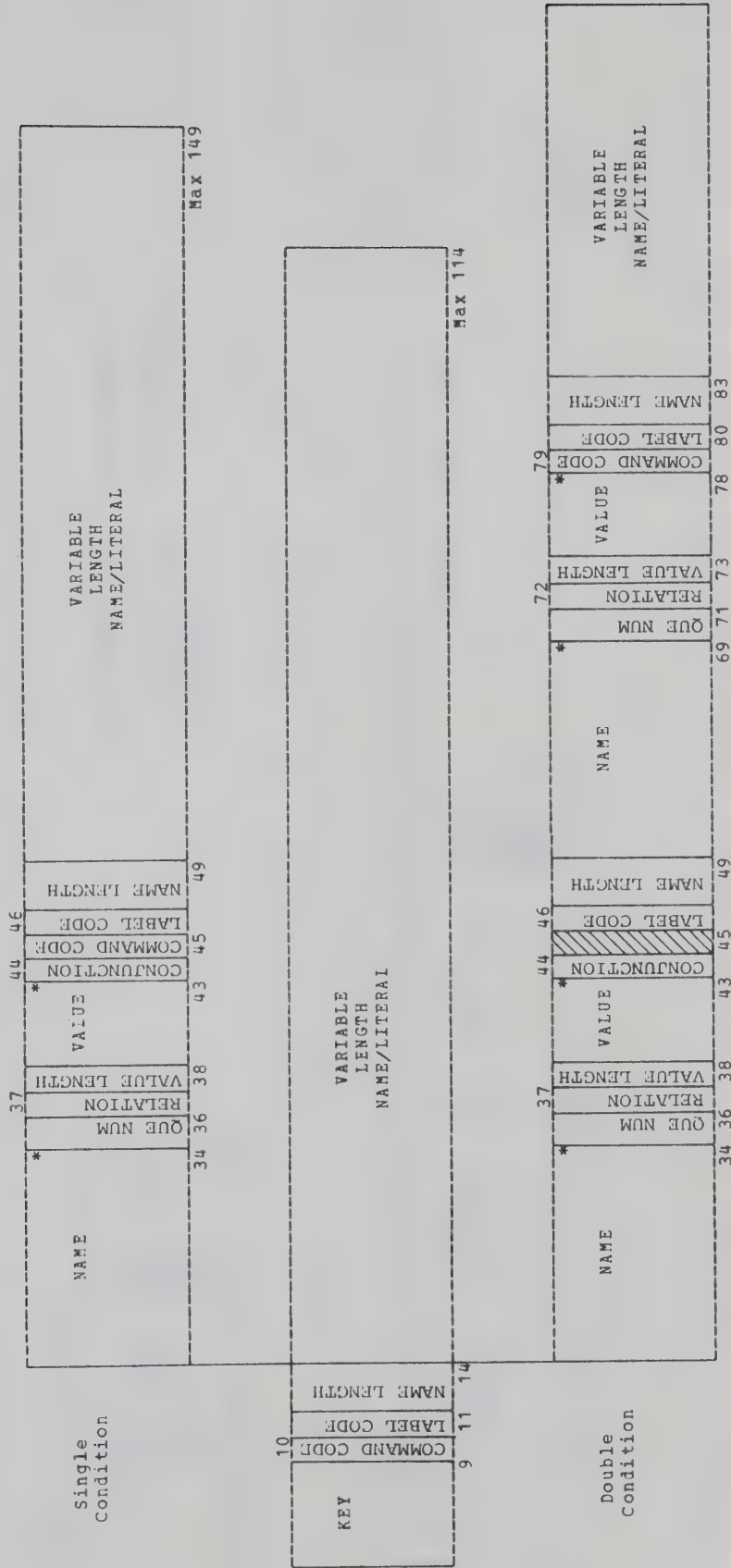
KEY	LOGIC REFERENCE.	LOGIC REFERENCE	----- up to 10 Logic Names per record -----
9	29	49	Max 209

TEST-KEY RECORD

KEY	TEST WEIGHT	NUM QUES	QUE #1	QUE #2	QUE #3	----- up to 20 questions per record -----
9	10	12	22	32	32	Max 212

ENCODED LOGIC RECORD

(* denotes variable length data within a fixed length field)



Command Codes

- 1 - GOTO
- 2 - IF
- 3 - MSG
- 4 - SHOW
- 5 - TEST
- 6 - WHEN

Label Codes

- 1 - Logic Name
- 2 - Display Name
- 3 - Test Name
- 4 - Literal

Relation Codes

- 1 - EQ
- 2 - NE
- 3 - LT
- 4 - LE
- 5 - GT
- 6 - GE

Conjunction Codes

- 1 - Single Condition
- 2 - Double Condition "AND"
- 3 - Double Condition "OR"

NOTE INFORMATION RECFD

KEY	NUM RECORDS	NOTE BASE KEY	NUM BATCH RECORDS	BATCH BASE KEY
9	13	22	26	35

Note_Codes

1 - Not Used
2 - Display Label
3 - Not Used
4 - Literal

Note_Status

0 - Pending
1 - Not Pending

NOTE RECORD

KEY	NEXT NOTE POINTER	STUDENT IDENTIFIER	USER IDENTIFIER	STATUS	NOTE CODE	NOTE LENGTH	VARIABLE LENGTH NOTE
9	18	38	58	60	63		Max 172

BATCH RECORD

KEY	USER IDENTIFIER	RELATIVE RECORD NUMBER	VARIABLE LENGTH "TAIM" COMMAND
9	29	35	Max 255

STATISTICS HEADER RECORD

KEY	NUM RECORDS	STATISTICS BASE
9	13	22

STATISTICS RECORD

KEY	USER/STUDENT IDENTIFICATION	DATE	TIME	BATCH/TERM	CPU TIME 1/1000 SECS	ELAPSE TIME OR PRINT LINES	VMSIZE	COST	CURRENT MODE	COMMAND #	RETURN CODE	VARIABLE LENGTH "TAIL" COMMAND OR INTERPRETED LOGIC STATEMENT
9	29	37	45	46	51	56	61	66	67	69	72	Max 255

HOLD INFORMATION RECORD

KEY	NUM HOLD RECORDS	HOLD BASE KEY
9	13	22

HOLD RECORD

KEY	STUDENT IDENTIFICATION	USER IDENTIFICATION	HOLD STATUS	DATE HELD	NUM DAYS
9	29	49	50	58	60

HOLD STATUS
0 - Holding
1 - Not Holding

APPENDIX D

SAMPLE SYSTEM FILE

SYSTEM FILE HEADER RECORD

<u>BYTES</u>	<u>BASE KEY FOR</u>	<u>VALUE</u>	<u>SECTION ON PAGE</u>
1 - 9		+00000000	
10 - 18	Command Section	+01000000	147
19 - 27	Logic Section	+02000000	148
28 - 36	Display Section	+03000000	149
37 - 45	Test Section	+04000000	149
46 - 54	Student Section	+05000000	149
55 - 63	User Section	+06000000	149
64 - 72	Hold Section	+07000000	150
73 - 81	Note Section	+08000000	151
82 - 90	Error Section	+09000000	152
91 - 99	Online Section	+10000000	158
100 - 108	Offline Section	+11000000	158

COMMAND_SECTION

+01000000043 TAIM COMMAND TABLE

Section Header
Record

+01001000AVERAGE	0411041
+01002000DISPLAY	0111007
+01003000HOLD	0411003
+01004000MARK	0211004
+01005000MODE	0411005
+01006000NOTE	0111001
+01007000POINT	0511006
+01008000QUIT	0111013
+01009000READ	0411002
+01010000REGISTER	0311008
+01011000TRACE	0111010
+01012000UNHOLD	0311037
+01013000UNNOTE	0311036
+01014000UNREGISTER	0511009
+01015000WHEN	0410042
+01016000WHO	0311011
+01017000WHY	0111012
+01018000CATALOG	0321043
+01019000DROP	0421016
+01020000EDIT	0221017
+01021000EVALUATE	0421018
+01022000EXECUTE	0420015
+01023000FIND	0121019
+01024000LABEL	0121021
+01025000MODE	0421022
+01026000PLACE	0221023
+01027000QUIT	0121028
+01028000RENAME	0620014
+01029000SAVE	0421026
+01030000SETKEY	0321027
+01031000WHY	0121024
+01032000CONDENSE	0430040
+01033000DISPLAY	0431033
+01034000DUMP	0430029
+01035000INITIALIZE	0431035
+01036000MODE	0431032
+01037000MTS	0331039
+01038000OFFLINE	0331038
+01039000QUIT	0131034
+01040000REGISTER	0331030
+01041000SET	0331020
+01042000UNREGISTER	0531031
+01043000WHY	0131025

43
Command Section
Records

LOGIC_SECTION

+02000000009 LOGIC FILE INITIALIZATION RECORDS

+02001000+00000000+00001000+00001000+00001001+00001002-99999999+01000000 File Header
Record

+02002000+00001000+00001001-99999999TAIM.START Index Records
+02003000+00001001-99999999+00001000TAIM.STOP +01000000
+01010000

+02004000+01000000+010010000001+0100500000010001+010090000000 } TAIM.START
+02005000+0100100011009TAIM.STOP
+02006000+01005000TAIM.STOP

+02007000+01010000+010110000001+0101500000010001+010190000000 Unit Leader Record
+02008000+010100011010TAIM.START Encoded Logic Record
+02009000+01015000TAIM.START References Record

DISPLAY_SECTION

+03000000001 DISPLAY FILE INITIALIZATION RECORD
+03001000+000000000+00001000-99999999-99999999-99999999+01000000 File Header
Record

TEST_SECTION

+04000000001 TEST FILE INITIALIZATION RECORD
+04001000+000000000+00001000-99999999-99999999-99999999+01000000 File Header
Record

STUDENT_SECTION

+05000000001 STUDENT FILE INITIALIZATION RECORD
+05001000+000000000+00001000-99999999-99999999-99999999+00100000 File Header
Record

USER_SECTION

+06000000002 USER FILE INITIALIZATION RECORDS
+06001000+000000000+00001000-99999999-99999999-99999999+00100000 File Header
Record
+06002000+001000000000+00101000 Statistics Header Record

NOTE SECTION

+08000000028 NOTE FILE INITIALIZATION RECORDS

+08001000+0000000-999999999ALH
+08002000+0001000-999999999A
+08003000+0002000-999999999B
+08004000+0003000-999999999C
+08005000+0004000-999999999D
+08006000+0005000-999999999E
+08007000+0006000-999999999F
+08008000+0007000-999999999G
+08009000+0008000-999999999H
+08010000+0009000-999999999I
+08011000+0010000-999999999J
+08012000+0011000-999999999K
+08013000+0012000-999999999L
+08014000+0013000-999999999M
+08015000+0014000-999999999N
+08016000+0015000-999999999O
+08017000+0016000-999999999P
+08018000+0017000-999999999Q
+08019000+0018000-999999999R
+08020000+0019000-999999999S
+08021000+0020000-999999999T
+08022000+0021000-999999999U
+08023000+0022000-999999999V
+08024000+0023000-999999999W
+08025000+0024000-999999999X
+08026000+0025000-999999999Y
+08027000+0026000-999999999Z

All/Class
Note Records

$$+08028000+00100000000+01000000000+02000000$$

Note Information Record

ERROR_SECTION

+09000000208 ERROR MESSAGE RECORDS

+09001005INSUFFICIENT PARAMETERS

+09001010ENCOUNTERED MORE THAN 2 PARAMETERS

+09001015DISPLAY LABEL PREFIX NOT "D-"

+09001020DISPLAY LABEL CONTAINS PREFIX ONLY

+09001025DISPLAY NAME EXCEEDS 20 CHARACTERS

+09001030A NULL RESPONSE TERMINATED THE COMMAND

+09001105FIRST CHARACTER OF DISPLAY NAME NOT ALPHABETIC

+09001110DISPLAY NAME NOT FOUND

+09002010NO PARAMETERS ARE PERMITTED FOR THIS COMMAND

+09003005INSUFFICIENT PARAMETERS

+09003010MORE THAN 2 PARAMETERS WERE ENCOUNTERED

+09003015NUMBER-OF-DAYS EXCEEDS 2 CHARACTERS

+09003020NUMBER-OF-DAYS CONTAINS A NON NUMERIC CHARACTER

+09003025STUDENTS MAY NOT BE HELD FOR ZERO DAYS

+09003030STUDENTS MAY NOT BE HELD FOR MORE THAN 15 DAYS

+09004005INSUFFICIENT PARAMETERS

+09004010MORE THAN 3 PARAMETERS WERE DETECTED

+09004015QUESTIONABLE "FULL" OPTION

+09004020LABEL PREFIX NOT "T-" OR "L-"

+09004025SPECIFIED LABEL IS LESS THAN 3 CHARACTERS

+09004030LABEL NAME EXCEEDS 20 CHARACTERS

+09005005INSUFFICIENT PARAMETERS

+09005010MORE THAN 1 PARAMETER WAS ENCOUNTERED

+09005015PARAMETER EXCEEDS 1 CHARACTER IN LENGTH

+09005020PARAMETER NOT "1", "2", "3" OR "?"

+09005025THIS USER NOT AUTHORIZED TO ENTER REQUESTED MODE

+09006005INSUFFICIENT PARAMETERS

+09006010MORE THAN 2 PARAMETERS WERE OBSERVED

+09006015LOGIC LABEL PREFIX NOT "L-"

+09006020LOGIC LABEL CONTAINS PREFIX ONLY

+09006025LOGIC NAME EXCEEDS 20 CHARACTERS

+09006105FIRST CHARACTER OF LOGIC NAME NOT ALPHABETIC
 +09006110LOGIC NAME NOT FOUND
 +09007005INSUFFICIENT PARAMETERS
 +09007010ENCOUNTERED MORE THAN 2 PARAMETERS
 +09007015SECOND PARAMETER NOT "HOLDS", "NOTES", "MESSAGES" OR "REGISTRY"
 +09008005INSUFFICIENT PARAMETERS
 +09008010FOUND MORE THAN 1 PARAMETER
 +09008015STUDENT ID IS NOT 6 CHARACTERS
 +09008020STUDENT ID CONTAINS A COMMA
 +09008025THREE CONSECUTIVE WRONG RESPONSES TERMINATED THE REGISTRATION
 +09008030A NULL RESPONSE TERMINATED THE REGISTRATION
 +09008035A "NO" RESPONSE CANCELLED THE REGISTRATION
 +09008105FIRST CHARACTER OF STUDENT ID NOT ALPHABETIC
 +09009005INSUFFICIENT PARAMETERS
 +09009010MORE THAN 1 PARAMETER WAS ENCOUNTERED
 +09009015STUDENT ID NOT 6 CHARACTERS
 +09009020STUDENT ID CONTAINS A COMMA
 +09009025THREE CONSECUTIVE WRONG RESPONSES TERMINATED THE COMMAND
 +09009030A NULL RESPONSE TERMINATED THE COMMAND
 +09009035A "NO" RESPONSE CANCELLED THE COMMAND
 +09009105FIRST CHARACTER OF STUDENT ID NOT ALPHABETIC
 +09009110STUDENT NOT REGISTERED
 +09010005INSUFFICIENT PARAMETERS
 +09010010MORE THAN 3 PARAMETERS WERE OBSERVED
 +09010015QUESTIONABLE "FULL" OPTION
 +09010020NUMBER-OF-DAYS EXCEEDS 3 CHARACTERS
 +09010025NUMBER-OF-DAYS CONTAINS A NON NUMERIC CHARACTER
 +09011005INSUFFICIENT PARAMETERS
 +09011010MORE THAN 3 PARAMETERS WERE OBSERVED
 +09011015QUESTIONABLE "MISSED" OPTION
 +09011020SPECIFIED LABEL IS LESS THAN 3 CHARACTERS
 +09011025LABEL PREFIX NOT "D-", "L-" OR "T-"
 +09011030LABEL NAME EXCEEDS 20 CHARACTERS
 +09011105FIRST CHARACTER OF LABEL NAME NOT ALPHABETIC
 +09016005INSUFFICIENT PARAMETERS
 +09016010MORE THAN 1 PARAMETER WAS ENCOUNTERED


```

+09016015SPECIFIED LABEL IS LESS THAN 3 CHARACTERS
+09016020LABEL NAME EXCEEDS 20 CHARACTERS
+09016025LABEL PREFIX NOT "L-", "D-" OR "T-"
+09016030"L-TAIM.START" AND "L-TAIM.STOP" CANNOT BE DROPPED
+09016035THE SPECIFIED LABEL NAME IS CURRENTLY REFERENCED BY LOGIC
+09016105FIRST CHARACTER OF LABEL NAME NOT ALPHABETIC
+09016110LABEL NAME DOES NOT EXIST
+09017005INSUFFICIENT PARAMETERS
+09017010MORE THAN 1 PARAMETER WAS DETECTED
+09017015WORKSPACE DESIGNATOR NOT 3 CHARACTERS IN LENGTH
+09017020FIRST 2 CHARACTERS OF WORKSPACE DESIGNATOR NOT "WS"
+09017025NO SUCH WORKSPACE
+09018005INSUFFICIENT PARAMETERS
+09018010MORE THAN 1 PARAMETER WAS ENCOUNTERED
+09018015WORKSPACE DESIGNATOR NOT 3 CHARACTERS IN LENGTH
+09018020FIRST 2 CHARACTERS OF WORKSPACE DESIGNATOR NOT "WS"
+09018025NO SUCH WORKSPACE
+09018030WORKSPACE IS NOT LABELLED
+09019005INSUFFICIENT PARAMETERS
+09019010MORE THAN 1 PARAMETER WAS OBSERVED
+09019015SPECIFIED LABEL IS LESS THAN 3 CHARACTERS
+09019020LABEL NAME EXCEEDS 20 CHARACTERS
+09019025LABEL PREFIX NOT "L-", "D-" OR "T-"
+09019105FIRST CHARACTER OF LABEL NAME NOT ALPHABETIC
+09019110LABEL NAME DOES NOT EXIST
+09020005INSUFFICIENT PARAMETERS
+09020010MORE THAN 2 PARAMETERS WERE OBSERVED
+09020015COMMAND# EXCEEDS 3 CHARACTERS IN LENGTH
+09020020COMMAND# CONTAINS A NON NUMERIC CHARACTER
+09020025SECOND PARAMETER NOT "ACTIVE" OR "INACTIVE"
+09020030NO SUCH COMMAND# --- CURRENT NUMBERS RANGE FROM 1 TO 43
+09020035COMMAND# TEMPORARILY NOT IN USE
+09021005INSUFFICIENT PARAMETERS
+09021010MORE THAN 2 PARAMETERS WERE DETECTED
+09021015WORKSPACE DESIGNATOR NOT 3 CHARACTERS IN LENGTH
+09021020FIRST 2 CHARACTERS OF WORKSPACE DESIGNATOR NOT "WS"

```


+09021025NO SUCH WORKSPACE
 +09021030SPECIFIED LABEL CONTAINS PREFIX ONLY
 +09021035LABEL NAME EXCEEDS 20 CHARACTERS
 +09021040LABEL PREFIX NOT "L-", "D-" OR "T-"
 +09021105FIRST CHARACTER OF LABEL NAME NOT ALPHABETIC
 +09021110LABEL NAME DOES NOT EXIST
 +09022005INSUFFICIENT PARAMETERS
 +09022010MORE THAN 1 PARAMETER WAS ENCOUNTERED
 +09022015PARAMETER EXCEEDS 1 CHARACTER IN LENGTH
 +09022020PARAMETER NOT "1", "2", "3" OR "?"
 +09022025THIS USER NOT AUTHORIZED TO ENTER REQUESTED MODE
 +09023005INSUFFICIENT PARAMETERS
 +09023010MORE THAN 3 PARAMETERS WERE DETECTED
 +09023015OBJECT WORKSPACE DESIGNATOR NOT 3 CHARACTERS IN LENGTH
 +09023020FIRST 2 CHARACTERS OF OBJECT WORKSPACE DESIGNATOR NOT "WS"
 +09023025NO SUCH OBJECT WORKSPACE
 +09023030NO SUCH SOURCE WORKSPACE
 +09023035SOURCE LABEL PARAMETER IS LESS THAN 3 CHARACTERS
 +09023040SOURCE LABEL NAME EXCEEDS 20 CHARACTERS
 +09023045SOURCE LABEL PREFIX NOT "L-", "D-" OR "T-"
 +09023105FIRST CHARACTER OF SOURCE LABEL NAME NOT ALPHABETIC
 +09023110SOURCE LABEL NAME DOES NOT EXIST
 +09026005INSUFFICIENT PARAMETERS
 +09026010MORE THAN 1 PARAMETER WAS ENCOUNTERED
 +09026015WORKSPACE DESIGNATOR NOT 3 CHARACTERS IN LENGTH
 +09026020FIRST 2 CHARACTERS OF WORKSPACE DESIGNATOR NOT "WS"
 +09026025NO SUCH WORKSPACE
 +09026030WORKSPACE NOT LABELLED
 +09026035ERRORS DETECTED DURING WORKSPACE EVALUATION
 +09026040FILE SPACE EXHAUSTED --- "QUIT" AND SEEK ASSISTANCE IMMEDIATELY
 +09026045A NULL WORKSPACE CANNOT BE SAVED
 +09027005INSUFFICIENT PARAMETERS
 +09027010MORE THAN 1 PARAMETER WAS OBSERVED
 +09027015WORKSPACE DESIGNATOR NOT 3 CHARACTERS IN LENGTH
 +09027020FIRST 2 CHARACTERS OF WORKSPACE DESIGNATOR NOT "WS"
 +09027025NO SUCH WORKSPACE


```

+09030005INSUFFICIENT PARAMETERS
+09030010MORE THAN 1 PARAMETER WAS OBSERVED
+09030015USER ID NOT 6 CHARACTERS IN LENGTH
+09030020THREE CONSECUTIVE WRONG RESPONSES TERMINATED THE REGISTRATION
+09030025A NULL RESPONSE TERMINATED THE REGISTRATION
+09030030A "NO" RESPONSE CANCELLED THE REGISTRATION
+09030105FIRST CHARACTER OF USER ID NOT ALPHABETIC
+09031005INSUFFICIENT PARAMETERS
+09031010MORE THAN 1 PARAMETER WAS DETECTED
+09031015USER ID NOT 6 CHARACTERS IN LENGTH
+09031020THREE CONSECUTIVE WRONG RESPONSES TERMINATED THE COMMAND
+09031025A NULL RESPONSE TERMINATED THE COMMAND
+09031030A "NO" RESPONSE CANCELLED THE COMMAND
+09031105FIRST CHARACTER OF USER ID NOT ALPHABETIC
+09031110USER NOT REGISTERED
+09032005INSUFFICIENT PARAMETERS
+09032010MORE THAN 1 PARAMETER WAS ENCOUNTERED
+09032015PARAMETER EXCEEDS 1 CHARACTER IN LENGTH
+09032020PARAMETER NOT "1", "2", "3" OR "?"
+09032025THIS USER NOT AUTHORIZED TO ENTER REQUESTED MODE
+09033005INSUFFICIENT PARAMETERS
+09033010ENCOUNTERED TOO MANY PARAMETERS
+09033015FIRST PARAMETER NOT "COMMANDS" OR "USERS"
+09033020MODE DESIGNATOR EXCEEDS 1 CHARACTER IN LENGTH
+09033025NO SUCH MODE
+09035005INSUFFICIENT PARAMETERS
+09035010MORE THAN 1 PARAMETER WAS DETECTED
+09035015FILENAME NOT "D-FILE", "H-FILE", "L-FILE", "N-FILE", "S-FILE", "T-FILE", "U-FILE"
+09036005INSUFFICIENT PARAMETERS
+09036010MORE THAN 2 PARAMETERS WERE OBSERVED
+09036020NOTE# EXCEEDS 3 CHARACTERS
+09036025NOTE# CONTAINS A NON NUMERIC CHARACTER
+09036030A NOTE# OF ZERO IS NOT PERMITTED
+09037005INSUFFICIENT PARAMETERS
+09037010MORE THAN 1 PARAMETER ENCOUNTERED
+09041005INSUFFICIENT PARAMETERS

```



```

+09041010MORE THAN 2 PARAMETERS ENCOUNTERED
+09041015QUESTIONABLE "FULL" OPTION
+09041020NO VALID TEST LABELS WERE ENTERED
+09043005INSUFFICIENT PARAMETERS
+09043010TOO MANY PARAMETERS GIVEN
+09043015FIRST PARAMETER NOT "D-FILE", "T-FILE", "L-FILE", OR "ALL".
+09043020SECOND PARAMETER NOT "XREF".
+09100205INVALID CLASS CODE
+09100210DOUBLE COMMA IN SIDS
+09100215FIRST CHARACTER OF SIDS IS COMMA
+09100220LAST CHARACTER OF SIDS IS COMMA
+09100225SIDS CONTAINS MORE THAN FIVE STUDENT IDENTIFIERS
+09100305STUDENT ID NOT 6 CHARACTERS
+091003101ST STUDENT ID NOT 6 CHARACTERS
+091003152ND STUDENT ID NOT 6 CHARACTERS
+091003203RD STUDENT ID NOT 6 CHARACTERS
+091003254TH STUDENT ID NOT 6 CHARACTERS
+091003305TH STUDENT ID NOT 6 CHARACTERS
+09100335STUDENT ID BEGINS WITH ILLEGAL CHARACTER
+091003401ST STUDENT ID BEGINS WITH ILLEGAL CHARACTER
+091003452ND STUDENT ID BEGINS WITH ILLEGAL CHARACTER
+091003503RD STUDENT ID BEGINS WITH ILLEGAL CHARACTER
+091003554TH STUDENT ID BEGINS WITH ILLEGAL CHARACTER
+091003605TH STUDENT ID BEGINS WITH ILLEGAL CHARACTER
+09100365STUDENT NOT REGISTERED
+091003701ST STUDENT NOT REGISTERED
+091003752ND STUDENT NOT REGISTERED
+091003803RD STUDENT NOT REGISTERED
+091003854TH STUDENT NOT REGISTERED
+091003905TH STUDENT NOT REGISTERED
+09100405COMMAND NOT YET IMPLEMENTED
+09100410COMMAND TEMPORARILY NOT AVAILABLE
+09100415COMMAND SUSPENDED VIA ATTENTION KEY

```


ONLINE_SECTION

```

+100000000006      COMMAND MONITOR LOGICAL I/O ASSIGNMENTS

+10001000TERMIN=*SOURCE* TERMOUT=*SINK*@UA(120) SCRATCH=-SCRATCH ATTNOUT=*MSINK*
+10002000LFILE=LOGIC DFILE=DISPLAY TFILE=TEST SFILE=STUDENT
+10003000UFILE=USER HFILE=HOLD NFILE=NOTE S$FILE=STUDENT
+10004000L$FILE=LOGIC D$FILE=DISPLAY T$FILE=TEST S$FILE=STUDENT
+10005000U$FILE=USER H$FILE=HOLD N$FILE=NOTE S$FILE=STUDENT
+10006000WS1=WS1 WS2=WS2 WS3=WS3 TWS1=-W$S$1 TWS2=-W$S$2 TWS3=-W$S$3

```

Logical File Name



Physical File Name

OFFLINE_SECTION

```

+110000000003      BATCH PROCESSOR LOGICAL I/O ASSIGNMENTS

+11001000LFILE=LOGIC DFILE=DISPLAY TFILE=TEST SFILE=STUDENT
+11002000HFILE=HOLD NFILE=NOTE UFILE=USER
+11003000MMSG=-MESGS OUTFILE=-PRINT BATCH=-BATCH STATS=*T*

```


APPENDIX E

MTS CONTROL STATEMENTS FOR USING TAIM

INVOKING THE INTERACTIVE TAIM SUBSYSTEM

```
$SIGNON CSID .....
password
$RUN TAIM
$SIGNOFF
```

INVOKING THE BATCH TAIM SUBSYSTEM

```
$SIGNON CSID .....
password
$SOURCE OFFLINE
$SIGNOFF
```

GENERATING THE TAIM SYSTEM

```
$SIGNON CSID .....
password
$COMMENT *** CREATE TAIM DATA FILES ***
$CREATE DISPLAY TYPE=LINE
$CREATE HOLD TYPE=LINE
$CREATE LOGIC TYPE=LINE
$CREATE NOTE TYPE=LINE
$CREATE STUDENT TYPE=LINE
$CREATE SYSTEM TYPE=LINE
$CREATE TEST TYPE=LINE
$CREATE USER TYPE=LINE
$COMMENT *** CREATE WORKSPACE FILES ***
$CREATE WS1
$CREATE WS2
$CREATE WS3
$COMMENT *** CREATE PROGRAM FILES ***
$CREATE BATCH TYPE=SEQ
$CREATE EXITA TYPE=SEQ
$CREATE EXITB TYPE=SEQ
$CREATE PRINT TYPE=SEQ
$CREATE TAIM TYPE=SEQ
$COMMENT *** CREATE BANNER PAGE ***
$CREATE BANNER
$COMMENT *** CREATE MTS CONTROL FILE ***
$CREATE OFFLINE TYPE=LINE
```



```

$COMMENT                                     *** INIT TAIM PROGRAM FILES ***
$SET IC=OFF
$COPY *SOURCE* TO BATCH
.
.
.   Batch TAIM Subsystem Object Modules
.
$CONTINUE WITH *PL1LIB RETURN
$ENDFILE
$COPY *SOURCE* TO EXITA
.
.
.   Sort Exit Object Module
.
$ENDFILE
$COPY *SOURCE* TO EXITB
.
.
.   Sort Exit Object Module
.
$ENDFILE
$COPY *SOURCE* TO PRINT
.
.
.   Message Printer Object Module
.
$ENDFILE
$COPY *SOURCE* TO TAIM
.
.
.   Interactive TAIM Subsystem Object Modules
.
$CONTINUE WITH *PL1LIB RETURN
$ENDFILE
$COMMENT                                     *** GENERATE SYSTEM FILE ***
$RUN *SOURCE* 5=*SOURCE* 6=SYSTEM
.
.
.   System File Generator Object Module
.
$ENDFILE
.
.
.   System File Generation Data Cards
.
$ENDFILE

```



```

$COMMENT                                     *** INIT BANNER PAGE ***
$GET BANNER
$NUMBER
.
.
.   Banner Page Data Cards
.
.
$UNNUMBER
$COMMENT                                     *** INIT MTS CONTROL FILE ***
$GET OFFLINE
$NUMBER
$$COMMENT                                     *** CREATE TEMPORARY FILES ***
$$CREATE -PRINT TYPE=SEQ SIZE=10P
$$CREATE -MESGS TYPE=SEQ SIZE=10P
$$CREATE -BATCH TYPE=SEQ SIZE=10P
$$CREATE -TEMPS TYPE=SEQ SIZE=10P
$COMMENT                                     *** MOUNT HISTROY TAPE ***
$$RUN *MOUNT PAR=XXXX *T* RING=IN POSN=*EOT* VOL=XXXXXX
$CONTROL *T* BSF
$COMMENT                                     *** START BATCH PROCESSOR ***
$$RUN BATCH
$COMMENT                                     *** DISMOUNT HISTORY TAPE ***
$$RUN *DISMOUNT PAR=*T*
$COMMENT                                     *** SORT TEACHER MESSAGES ***
$$RUN *SORT
SORT=CH;A;1;42
INPUT=-MESGS;U;142   OUTPUT=-TEMPS;U;142
MNR=999
$$ENDFILE
$COMMENT                                     *** PRINT TEACHER MESSAGES ***
$$RUN PRINT 5=-TEMPS 6=*SINK*
$COMMENT                                     *** SORT AND PRINT LESSONS ***
$$RUN *SORT+EXITA   SPRINT=*SINK*
SORT=CH;A;1;7
INPUT=-PRINT;U;140   OUTPUT=*DUMMY*;U;133
MNR=9999
$$ENDFILE
$COMMENT                                     *** SORT BATCHED COMMANDS ***
$EMPTY -TEMPS
$$RUN *SORT+EXITB   SPUNCH=-TEMPS
SORT=CH;A;1;26
INPUT=-BATCH;U;126   OUTPUT=*DUMMY*;U;100
MNR=999
$$ENDFILE
$COMMENT                                     *** START COMMAND MONITOR ***
$$RUN TAIM
$$CONTINUE WITH -TEMPS RETURN
$ENDFILE
$UNNUMBER

```


APPENDIX F

TAIM MODULE DESCRIPTIONS

MODULES COMPRISING THE BATCH TAIM SUBSYSTEM

1 of 1

Module Name	Source Code	Total Source	Size In Bytes	MTS File Name	Description
CONTROL	A	397	1,452	BATCH	localizes operating system dependencies
DRIVER	P	458	12,584	BATCH	controls student processing; accumulates statistical data; spools batched commands; performs file maintenance
HASHNIT	P	48	1,478	BATCH	initializes directory index tables
INTERP	P	392	10,322	BATCH	interprets Lesson Logic Statements
MAIN	F	125	3,428	PRINT	summarizes and prints sorted teacher messages
PRINTER	P	262	9,082	BATCH	spools student lessons and teacher messages
SEARCH	P	50	1,414	BATCH	searches a specified directory for an item
SORTE3	F	9	292	EXITA	removes first 7 bytes of each sorted record
SORTE3	F	9	292	EXITB	removes first 26 bytes of each sorted record
STATIST	P	89	3,228	BATCH	generates student statistics
Totals		1,839	43,572		A=Assembler F=Fortran P=PL/1

MODULES COMPRISING THE INTERACTIVE TAIM SUBSYSTEM (MTS file TAIM)

1 of 3

<u>Module Name</u>	<u>Source Code</u>	<u>Total Source</u>	<u>Size In Bytes</u>	<u>Associative Command Numbers</u>	<u>Description</u>
AVERAGE	P	303	7,558	41	executes AVERAGE command
BANNER	P	32	1,386		prints banner page
BATCHER	P	142	4,602	38	executes OFFLINE command; places interrupted commands into the overnight batch queue
CATALOG	P	167	4,401	43	executes CATALOG command
CONTROL	A	397	1,452		localizes operating system dependencies
DECIDE	P	264	7,062		interprets Decision Logic Statements
DECODE	P	140	6,626		translates encoded Logic Statements into a textual representation
DELETE	P	75	2,498		deletes an item from a specified directory
DESOLVE	P	76	2,273		deletes a Day name from a reference list
DSPLAY1	P	244	8,094	07	executes DISPLAY command (Mode 1)
DSPLAY3	P	127	4,294	33	executes DISPLAY command (Mode 3)
ENCODE	P	432	12,902		translates textual Logic Statements into an encoded representation
EVALSAV	P	230	7,190	18,26	executes EVALUATE and SAVE commands
EXTRACT	P	35	1,130		initializes input buffer component table

MODULES COMPRISING THE INTERACTIVE TAIM SUBSYSTEM (MTS file TAIM)

2 of 3

<u>Module</u> <u>Name</u>	<u>Source</u> <u>Code</u>	<u>Total</u> <u>Source</u>	<u>Size</u> <u>In</u> <u>Bytes</u>	<u>Associative</u> <u>Command</u> <u>Numbers</u>	<u>Description</u>
FINDROP	P	189	4,690	16,19	executes FIND and DROP commands
GETBUF	P	51	1,382		centralized read operation; removes leading and trailing blanks from an input buffer
HASHNIT	P	47	1,510		initializes directory index tables
HOLD	P	161	3,209	03,37	executes HOLD and UNHOLD commands
INITAL	P	83	2,846	35	executes INITIALIZE command
INSERT	P	110	3,206		inserts an item into a specified directory
LABEDIT	P	118	3,102	17,21	executes LABEL and EDIT commands
MARK	P	261	6,926	04	executes MARK command
MODE	P	57	1,542	5,22,32	executes MODE command (Mode 1,2,3)
MONITOR	P	553	17,568		acknowledges user; accepts user commands; activates command work modules; generates user statistics; handles attention traps
MTS	P	47	2,142	39	executes MTS command
NOTE	P	265	6,382	01,36	executes NOTE and UNNOTE commands
PLACE	P	197	4,923	23	executes PLACE command
POINTER	P	114	2,882	06	executes POINT command

MODULES COMPRISING THE INTERACTIVE TAIM SUBSYSTEM (MTS file TAIM)

3 of 3

<u>Module</u> <u>Name</u>	<u>Source</u> <u>Code</u>	<u>Total</u> <u>Source</u>	<u>Size</u> <u>In</u> <u>Bytes</u>	<u>Associative</u> <u>Command</u> <u>Numbers</u>	<u>Description</u>
READER	P	236	6,750	02	executes READ command
RESOLVE	P	114	3,409		inserts a Day name into a reference list
SEARCH	P	53	1,474		searches a specified directory for an item
SET	P	102	2,590	20	executes SET command
SETKEY	P	291	11,034	27	executes SETKEY command
SIDSCAN	P	119	3,414		parses any sid parameters
STUREG	P	369	12,110	08,09	executes REGISTER command (Mode 1)
TRACE	P	252	7,478	10	executes TRACE command
UNIQUE	P	109	3,498		determines which Day names are to be resolved or desolved
USEREG	P	208	6,722	30,31	executes REGISTER command (Mode 3)
WHO	P	195	4,378	11	executes WHO command
WHY	P	48	1,811	12,24,25	executes WHY command (Mode 1,2,3)
Totals		7,013	198,446		A=Assembler F=Fortran P=PL/1

B30070